Universidad Carlos III de Madrid

Escuela Politécnica Superior - Leganés

Ingeniería Telemática

Trabajo Fin de Grado

Interface management framework using context information from mobile devices

AUTOR: Raúl Gómez Moreno
TUTOR: Jose Ignacio Moreno Novella

Leganés, 2016

TITULO: *Interface management framework using context information from mobile devices*

AUTOR: Raúl Gómez Moreno

TUTOR: Jose Ignacio Moreno Novella

La defensa del presente Proyecto Fin de Carrera se realizó el día 7 de Octubre de 2016 siendo calificada por el siguiente tribunal:

PRESIDENTE: David Larrabeiti López

SECRETARIO: Rodolfo Cuerno Rejado

VOCAL: Yoel Gustavo Yera Mora

Habiendo obtenido la siguiente calificación:

CALIFICACIÓN:

Presidente **Secretario** **Vocal**

Agradecimientos

En primer lugar agradecer a mi tutor Jose Ignacio la oportunidad que me dió ofreciéndome estas prácticas Erasmus+ donde pude realizar el presente trabajo y el apoyo tanto técnico como moral y personal que me brindó antes, durante y después de mi estancia en Berlín.

A mi familia y amigos, especialmente a mis padres y mi hermana, por haber sido el pilar fundamental que me apoyó desde que comencé mis estudios y que siempre han aguantado todos mi cabreos, frustraciones y agobios.

A pesar de que todos los compañeros han sido un apoyo, mencionar a Jorge, Dani y Cristian, que más que compañeros han sido amigos desde que empezamos hasta que hemos acabado, gracias por todos los momentos que hemos pasado juntos. Ahora puedo decir que ha sido todo un placer trabajar a vuestro lado.

Por último me gustaría ser algo más genérico y me gustaría agradecer a la Universidad Carlos III de Madrid toda la ayuda que me ha ofrecido hasta ahora, desde las becas que me han permitido continuar estudiando como de los premios de reconocimiento que me han dado las fuerzas necesarias para seguir esforzándome y superándome día a día.

Resumen

Mientras trabajamos en la oficina, cuando estamos en nuestro hogar descansando o incluso cuando vamos a hacer la compra semanal, nuestros dispositivos móviles, ya sean tablets, móviles o wereables detectan un gran abanico de redes inalámbricas a las que poder conectarnos para no utilizar la tarifa de datos que tenemos contratada con nuestro operador.

Si somos un poco curiosos, quizás nos habremos hecho preguntas como: ¿A qué red de todas las disponibles me conviene conectarme? ¿Con cuál de ellas tendré un ancho de banda mayor? Si me conecto a una determinada red, ¿estarán a salvo los datos que envie durante mi conexión?

A su vez, si nos posicionamos como operadores que brindan la infraestructura que permiten que estas redes funcionen podremos pensar: ¿Estoy ofreciendo al usuario el servicio adecuado? ¿Cuántos de los equipos que detectan mi red acaban estableciendo una conexión? ¿Qué tipo de dispositivos utilizan mi red? ¿La pérdida de potencia por propagación de la señal es muy elevada? ¿Debo mejorar mi infraestructura?

Con este trabajo se pretende implementar una interfaz de usuario o aplicación para cualquier dispositivo Android que permita utilizar la información que nuestros dispositivos móviles son capaces de detectar sobre las redes inalámbricas exisitentes a nuestro alrededor para dar respuesta a todas las preguntas planteadas anteriormente.

Abstract

While we are working, when we are home resting or even when we go to the supermarket, our mobiles devices (phones, tablets and other wereables) recognize a big variety of wireless networks which can be used insetead of use our mobile data plan.

If we are a little bit curious, maybe we asked ourselves questions like: What is the best wireless network to connect? What wireless network will offer me more bandwidth? If I am connected to this network, my personal data will be safe?

In turn, if we place ourselves like a network company which has the network infrastructure, we could think about: Am I offering the best service to my clients? How many devices are using my network? Which type of devices are using my network? Signal loss is too much bigger? Should I improve my service?

Within this project we are going to develop an Android application which allow us to use information which our mobiles devices can be detect about wireless networks to answer all the questions posed before.

Índice general

Lista de Figuras

Lista de Tablas

Chapter 1

Introduction

In this chapter we will see what was the problem and what was the solution which we used to solve it showing the project phases and the environment used. Also we explain how this document is structured.

1.1. Motivation

Please, feel like a new employeer that a company has just to recruit. Company's scretary give you the "welcome kit"which came with an Android mobile device and an Android Tablet (both of them totally new, without any previous configuration). While you are waiting for your job tasks, you decide starting your devices internet configuration. After turn on your wireless adapter, you can see that there are too much wireless networks, some with the same name, others without encryption but with a strong signal and others which are using 5 Ghz frecuency. So now you probably will ask yourself: Which one will be the best to keep safe my sent data? With which one I will have more bandwith? All of them comply security company policies?

Now, fell like a technical support of a netowork company which provides internet services to private corporations offices.

Human resource department have just to hire new employees and you have to reconfigure the wireless network to get more eficiency and client satisfaction. Due to the big area which cover all the offices, when the build was built, the support manager decided to create more than one access point located in different places indentifing users and uses.

As I said before, now you have to reconfigure that configuration so you will need to check if the users that are using a determinate access point are allowed, if all the user which are using the same access point are receiving the correct signal strength or should use another one, if the signal loss is not

too much bigger or for example if there are too much people that are using the same network.

How would you get all this information?

1.2. Goals and project phases

Within this project we are going to develop an user interface which allow us to get all the information about wireless networks that are living around us to use with all Android not rooted[1] devices.

Furthermore, this information will be presented textually and graphically and also we will have the possibility to send that information to a server which store and analyze the information in order to improve the service which a network operator offers.

To check if our application is working as we think, we will need to have an Android mobile device, more than one wireless networks and a server[2]. In our case, we used a local server located in our personal computer.

To get the goal of this project, the tasks were the following:

- Investigation of information available on mobile devices.
- Implementation of a hierarchical context collection system which allows to efficiently collect the relevant context information.
- Implementation of an interface management system.
- Detailed performance investigations, analysis of the results, final report, and demonstration guide.

Moreover, the following phases were defined:

- Got familiar with the already available implementations and has an overview about context information provided by mobile devices.
- First version of the context collection system implementation and the interface management system implementation.

[1]Be root allows the user to have some privileges which bring the possibility to break the limits imposed by the manufacturer to perform actions like replacing system applications, run special software or improve device performance.

[2]Server structure and how it analyze the data that we send from our mobile device are not the focus of this project because they are other projects that were running in parallel by other team from T-Labs.

- Stable version of the context collection system implementation and the interface management system implementation.
- Performance evaluation measurements and final report.

1.3. Development environment project

This project was carried out during Erasmus Placement program from 1 of September 2015 to 1 of March 2016 in Seamless Network Control Department of Deutsche Telekom Innovation Laboratories (T-Labs, Berlín) under the supervision of Dr. Nico Bayer with the aim of providing current research with an user interface which collect and analyse information about wireless networks in order to improve the user experience and in the other hand, improve the infraestructure which operators like Deutsche Telekom is offering actually.

T-Labs, as unit of products and innovation under Deutsche Telekom organization carries out projects in all areas of Deutsche Telekom group. At the same time, T-Labs is associated with Technology Institute of Berlin (TU Berlín) wich means that T-Labs is a privately organized entity which is closely integrated with teaching activities and research conducted.

1.4. Structure of this document

This book is organized as show below:

- Chapter 1 introduces the reader to the posed problem showing how is possible to solve it developing our application.
- Chapter 2 explains us all the technical knowledges that are necessary to understand the project and the study which was done before the project started.
- Chapter 3 shows the architecture design of the entire system, ie, the different elements that took part in the solution proposed. Also, this chapter shows the graphical interface, internal architecture and the process of information exchange which was used to send information to the server.
- Chapter 4 draws conclusions that were obtained after finish the project and provides orientations for future work lines.
- Annex A includes regulatory framework, task planning and project budget.

- Annex B is the summary of this project.

Capítulo 2

Estado del arte

En este capítulo se harán mención a los elementos conceptuales que han servido de base para la investigación y los estudios previos necesarios para realizar el presente trabajo.

El estudio previo que ha sido necesario para el desarrollo de este proyecto puede ser dividido en tres partes claramente diferenciadas: conocer qué es Android y cómo funciona, aprender y reforzar conocimientos sobre cómo funcionan las tecnologías de red inalámbrica y aprender a utilizar el software empleado.

En primer lugar, partiendo de la historia de Android, nos adentramos en lo más profundo de este explorando su arquitectura y viendo cómo han ido cambiando las versiones comercializadas hasta el día de hoy. Terminamos con un enfrentamiento de este sistema operativo con sus rivales.

A continuación se presentan las distintas tecnologías de red inalámbricas existentes centrándonos en las redes WLAN que serán las usadas en nuestro proyecto. Para este tipo de redes se hace un análisis de los elementos que las forman y se aclaran una serie de términos y conceptos que nos serán de gran utilidad para comprender cómo funcionan este tipo de redes.

Por último hacemos mención al software que hemos utilizado. Por un lado se presenta el entorno de desarrollo con el que hemos trabajado para el desarollo de nuestra aplicación y por otro presentamos la base de datos que se ha decidido usar para el almacenamiento de la información en el servidor dado.

2.1. Android

Android [1] es un sistema operativo basado en Linux, un núcleo de sistema operativo libre, gratuito y multiplataforma. Todo esto ha hecho que la mayoría de empresas presten su interés para su uso en los diferentes dispositivos comercializados.

El porcentaje de dispositivos que corren bajo este sistema operativo está aumentando significativamente y se espera que se convierta en el estándar de facto en la industria de las telecomunicaciones.

En esta sección veremos como se ha llegado hasta esta situación.

2.1.1. Historia

Este sistema operativo se creó bajo el nombre de Android Inc, una compañía que Andy Rubin, Nick Sears, Rich Miner y Chris White pusieron en marcha en la localidad de Palo Alto en el año 2003. El objetivo era crear un sistema operativo abierto que fuese capaz de generar una experiencia de usuario innovadora.

En 2005, Google muestra su interés tanto por la idea como por sus ingenierios y decide comprar Android Inc por la cantidad de 50 millones de dólares.

Google decidió en 2006 poner a trabajar a su equipo de ingenieros en un kernel linux para desarrollar una interfaz de usuario accesible y sencilla a la sombra de la prensa y demás fuentes de información.

La primera versión oficial de Android, **Android 1.0 Apple Pie**, fue lanzada en 2007 tras un acuerdo de colaboración con estándares abiertos entre LG, Qualcomm, Intel, T-Mobile, Sprint Nextel, Moto, HTC, Sony Ericsson, Vodafone y ZTE.

La premisa con la que se presentó esta versión de Android era la aparición de un sistema gratuito y de código abierto. [1]

Podríamos decir que el mercado se abrió con la venta del primer terminal con Android: HTC Dream. A partir de este momento Google fue cogiendo fuerza incorporando mejoras a sus versiones, ofreciendo a los usuarios más prestaciones y haciéndose así con prácticamente toda la couta de mercado (En 2015 Android estaba presente en el 81,5 % de todos los dispositivos vendidos).

[1]Software distrbuido y desarrollado libremente. Se focaliza más en los beneficios prácticos (acceso al código fuente) que en cuestiones éticas o de libertad que tanto se destacan en el software libre. "https://www.gpsos.es/soluciones-open-source/definicion-de-open-source/"

2.1.2. Arquitectura

La plataforma Android contiene una pila de software donde se incluye un sistema operativo (OS), middleware [2] y aplicaciones básicas para el usuario.

Podemos dividir la arquitectura en las siguientes capas:

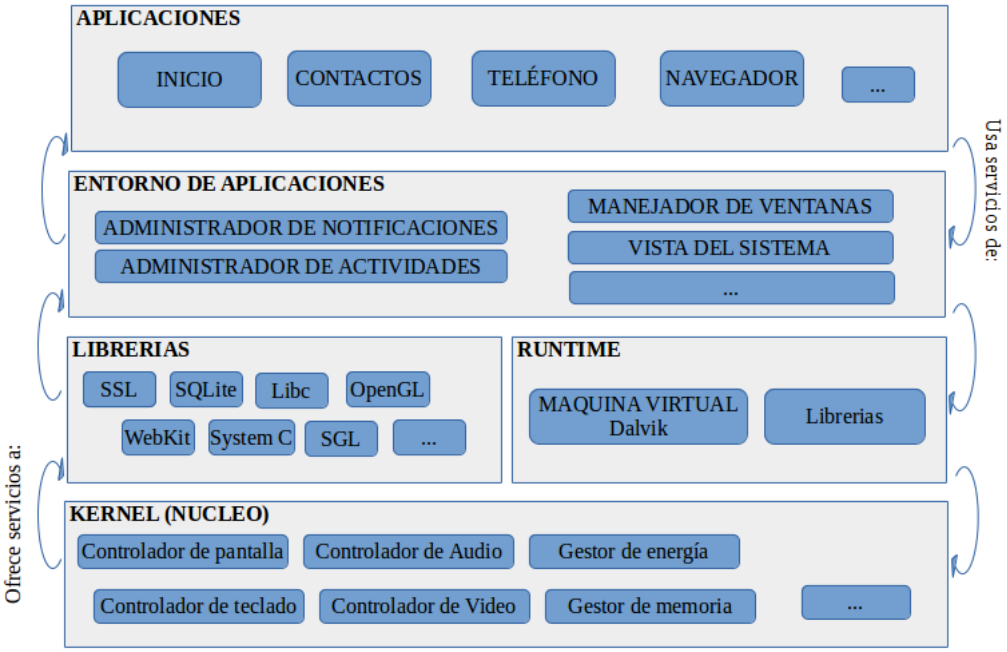

Figura 2.1: Arquitectura Android

Veamos ahora la función de cada una de estas capas:

- **Kernel**
  El núcleo de Android está formado por el sistema operativo Linux 2.6. Seguridad, manejo de memoria, tareas multiproceso, pila de protocolos y soporte de drivers son algunos de los servicios que esta capa ofrece a sus "superiores".

- **Librerías**
  Esta capa incluye todas las librerias en C/C++ utilizadas por cualquier componente del dispositivo. Destacar que todas las librerías están compiladas en código nativo del procesador y que la mayoría utilizan código abierto. Nos podemos encontrar con librerias como SGL (motor de gráficos 2D), FreeType (fuentes en bitmpa y renderizado vectorial), SQLite

[2] El middleware es un término que se utiliza para describir productos separados que sirven como unión entre dos aplicaciones de software de manera que puedan intercambiar datos. Las aplicaciones middleware se utiliza en entornos complejos y no representan necesariamente interacciones uno a uno entre los agentes de software.

(motor de base de datos), SSL (servicios de encriptación), Librerías 3D (proyección 3D, aceleradores de hardware), Media Framework (soporte de codecs de reproducción y grabación como MP3, AAC, JPG, PNG, etc), etc.

- **Runtime**
  Debido a la poca memoria y capacidad de procesamiento de los dispositivos no fue posible utilizar la maquina virtual que se usa en Java, por ello Google tomó la decisión de crear la maquina virtual Dalvik.

  Esta máquina virtual intenta optimitzar los recursos para conseguir el mayor rendimiento posible basando su funcionamiento en registros. Cada aplicación corre en su propio proceso con su propia instancia de la maquina virtual Davilk delegando al kernel algunas funciones como el manejo de memoria a bajo nivel.

  Gracias a mejoras de software y hardware, a partir de Android 5.0, ART reemplaza a Davilk. A diferencia de Davilk, que desde Android 2.2 utiliza "just in time"para compilar el código cada vez que se inicia una aplicación, ART introduce el uso de AOT que crea un archivo de compilación posterior a la instalación de la aplicación. Este archivo es utilizado al abrir la aplicación, evitando así el uso del procesador para compilar continuamente y consiguiendo así una notable mejora en el consumo de batería. Para que exista compatibilidad con versiones anteriores, ART utiliza el mismo código de bytes de entrada que Davilk, el cual es suministrado a través de archivos .dex como parte de los archivos APK.

- **Entorno de aplicaciones**
  Representa todas aquellas herramientas para el desarrollo de cualquier aplicación ya sean las propias del dispositivos, las desarrolladas por Google o incluso las que el propio usuario crea. Entre la multitud de API's [3] existentes podemos destacar las siguientes:

  - **Administrador de notificaciones (Notification Manager)**
    Permite mostrar alertas personalizadas en la barra de estado. Mediante estas notificaciones las aplicaciones comunican al usuario un determinado evento que ha ocurrido durante su ejecución: una llamada recibida, un mensaje recibido o por ejemplo un aviso del calendario.
  - **Manejador de ventanas (Window Manager)**
    Gestiona las ventanas de aplicaciones y usa la librería Surface Manager.

[3] Application Programming Interface (API) es un conjunto de reglas y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas.

- **Vista del sistema (View System)**
  Proporciona los elementos necesarios para poder construir interfaces de usuario (GUI) como botones, listas, mosaicos, etc.
- **Administrador de localización (Location Manager)**
  Dota a las aplicaciones con la posibilidad de conocer la localización y el posicionamiento actual del dispositivo.
- **Servicio XMPP (XMPP Service)**
  Colección de API para utilizar este protocolo de intercambio de mensajes basado en XML.
- **Manejador de actividades (Activity Manager)**
  Controla el ciclo de vida de las aplicaciones proporcionando además un sistema de navegación entre ellas.
- **Proveedor de contenidos (Content Providers)**
  Mecanismo para acceder a datos de otras aplicaciones como son direcciones de correos, números de teléfono, fotografías, etc.

- **Aplicaciones**
  En este nivel podemos incluir todas las aplicaciones que vienen preinstaladas con el sistema operativo y las que el usuario añade posteriormente. Cada una de estas aplicaciones utilizará los servicios que le ofrece la capa inferior.

2.1.3. Versiones

Cada versión de Android tiene asociado un nombre de dulce que es como habitualmente se le conoce al software de modo general. Además, las iniciales de los nombres de cada una de las versiones siguen un orden alfabético como vamos a ver a continuación. Según Google:

"La razón por la que todas las versiones de Android traen nombres de dulces es porque los smartphones y tablets endulzan nuestra vida y, entre estos, el chocolate es particularmente adictivo, siendo KitKat nuestro favorito."

Hasta el día de hoy catorce han sido las distintas versiones que Android ha ofrecido al usuario. Echemos un vistazo a todas ellas analizando qué mejoras fueron añadiendo con respecto a las anteriores.

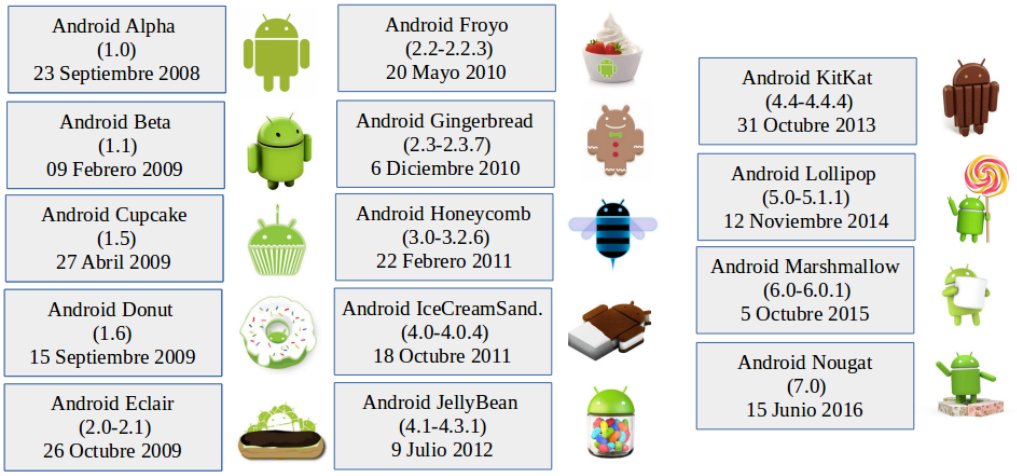

Figura 2.2: Versiones

- **Android Alpha (1.0)**

  Posteriormente conocido como Apple Pie se presentó como la primera versión con las siguientes funciones:

  - Notificaciones en menú desplegable
  - Widgets en el escritorio
  - Tienda de aplicaciones (Android Market)
  - Google Mail, Contacts, Calendar, Youtube y Maps
  - Soporte para cámara

- **Android Beta (1.1)**

  También conocido como Banana Bread apareció para solucionar muchos de los bugs y errores que presentaba la versión anterior.

- **Android Cupcake (1.5)**

  Se actualiza la versión del kernel a 2.6.27 con las siguientes actualizaciones:

  - Transiciones de pantalla
  - Conexión automática de Bluetooh
  - Soporte Bluetooth A2DP y AVRCP
  - Teclado con predicción de texto
  - Grabación y reproducción de vídeos con camcoder

- **Android Donut (1.6)**

  Basada en el kernel 2.6.29 se presentó con:

- Tienda de aplicaciones mejorada
- Cámara, grabación y galería de fotos integrados entre sí
- Selección multiple activada
- Búsqueda por voz
- Soporte de pantallas WVGA
- Mejoras de rendimiento
- Navegación por gestos

- **Android Eclair (2.0-2.1)**

  Esta vez se decide utilizar la misma versión del kernel para ofrecer mejoras como:

  - Mejora de la velocidad de hardware
  - Soporte HTML5
  - Soporte Microsoft Exchange
  - Zoom digital
  - Bluetooth 2.1
  - Fondos de pantalla animados
  - Flash en la cámara trasera
  - Interfaz de usuario más intuitiva

- **Android Froyo (2.2-2.2.3)**

  En 2010 se presenta esta versión que posteriormente sería una de las más conocidas debido a funcionalidades como:

  - Optimización del rendimiento del sistema operativo
  - Integración de JavaScript v8 en el navegador web
  - Soporte mejorado de Microsoft Exchange
  - Función hotspot Wifi (permite utilizar nuestro terminal como punto de acceso inalámbrico)
  - Soporte Adobe Flash
  - Marcación por voz
  - Múltiples teclados
  - Actualizaciones automáticas de las aplicaciones instaladas
  - Soporte para pantallas de alta resolución (720p)

- **Android Gingerbread (2.3-2.3.7)**

  A finales de año aparece la versión que sustituyó a Froyo introduciendo importantes mejoras como:

  - Rediseño de la interfaz de usuario
  - Decodificación AAC
  - Rediseño del teclado
  - Soporte para llamadas VoIP
  - Soporte para pantallas de mayores dimensiones
  - Mejoras de ecualización, virtualización y control de graves.
  - Aparecen las funciones cortar, copiar y pegar en el sistema.
  - Giroscopio y barómetro
  - Administrador de tareas y control de energía

- **Android Honeycomb (3.0-3.2.6)**

  Esta versión podemos decir que es un tanto especial ya que solo apareció para tablets y es quizás la menos conocida. A pesar de esto corrigió importantes errores con respecto a la anterior e introdujo la optmización para tablets:

  - Soporte para escritorio 3D con rediseño de widgets
  - Sistema multitarea
  - Mejoras en la navegación web: Navegación por pesatañas, navegación privada, etc
  - Mejora en soporte para redes Wifi
  - Compatibilidad con periféricos USB
  - Redimensión de aplicaciones

- **Android IceCreamSandwich (4.0-4.0.4)**

  Esta versión supuso un gran cambio en la historia de Android ya que llega con la unificación entre tablets y móviles.

  - Uso de Android en cualquier dispositivo
  - Mejora de interfaz
  - Botones virtuales
  - Gestor de consumo de datos móviles
  - Mejoras en el corrector de texto

- Aceleración por hardware
- Opción foto panorámica
- Soporte para lápices táctiles
- Reconocimiento facial
- Función compartir contenidos (Android Beam)
- Acceso rápido para realizar capturas de pantalla
- Posiblidad de borrar aplicaciones preinstaladas

- **Android JellyBean (4.1-4.3.1)**

  Otras de las grandes actualizaciones que Google presentó en Google I/O 2012 con los siguientes avances:

  - Project Butter par mejora de redimiento y gráficos
  - Servicio de voz inteligente de Google (Google Now)
  - Navegador Google Chrome
  - Dictado de voz sin conexión a internet
  - Mejoras relacionadas con los widgets y su redimensionamiento
  - Novedades en el diseño de las notificaciones de escritorio
  - Nuevo diseño del panel de control
  - Gestual Mode para el control del dispositivo para personas invidentes
  - Soporte con Miracast (streaming de audio y vídeo)

- **Android KitKat (4.4-4.4.4)**

  Una de las grandes apuestas y la favorita de Google que introdujo mejoras significativas como:

  - Mejoras en la cámara (velocidad enfoque, saturación, exposición, etc)
  - Aparición de Google Fotos para sustuir la Galería que venía preinstalada por defecto
  - Posibilidad de activar la maquina virtual ART
  - Solución a errores y bugs
  - Sonido uniforme para todas las aplicaciones

- **Android Lollipop (5.0-5.1.1)**

  Google comienza a seleccionar qué dispositivos actualizarán a esta versión y cuales no. Este comportamiento, en cierto medida, obliga al usuario a cambiar de dispositivo en caso de querer mantenerse acutualizado con la última versión de Android. Añade funcionalidades extra que no estaban en la versión anterior como son:

  - Tap and go (Basado en NFC y Bluetooth para transferir datos, configuración, aplicaciones y detalles de cuentas)
  - Aplicación Linterna (usando el flash led)
  - Notificaciones programables (se puede seleccionar de qué aplicación recibir notificaciones cómo hacerlo y cuándo)
  - Posibilidad de manejar las notificaciones con la pantalla bloqueada
  - Nuevo gestor de batería que permite conocer el uso y el tiempo restante de esta
  - Modo invitado para evitar que un intruso pueda acceder a nuestros datos cuando esta usando nuestro terminal
  - Dispositivos de confianza (dentro del rango de conectividad no es necesario desbloquear el dispositivo)
  - Función no molestar
  - Opción Ahorro de batería que optimiza el sistema para aumentar la autonomía de la batería mientras la opción se encuentra activada

- **Android Marshmallow (6.0-6.0.1)**

  Google introduce casi 50 funcionalidades nuevas como:

  - Google Now desde la pantalla de bloqueo
  - Posibilidad de desinstalar aplicaciones desde el escritorio
  - Información extra de aplicaciones en la barra de estado
  - Nuevo diseño del cajón de aplicaciones
  - Posibilidad de personalizar los accesos rápidos
  - Posibilidad de compartir o eliminar una captura desde la barra de estado
  - Reorganización de teclado para tablets
  - Nueva interfaz en los ajustes
  - Control de permisos granular (para cada aplicación)
  - Optimización de la batería

- Posibilidad de elegir la banda de la zona Wifi
- Menu de impresión
- Pago con Android Pay (basado en NFC y Host Card Emulation)
- Desbloqueo por huella dactilar

- **Android Nougat (7.0)**

  Actualmente se encuentra en versión beta[4]. Algunas de las mejoras que promete son:

  - Mutilventana nativa
  - Mejora de notificaciones
  - Sugerencias en configuraciones
  - Mejoras en las configuraciones rápidas
  - Instant apps (usar aplicaciones sin tener que descargarlas e instalarlas)
  - Ahorro de datos
  - Mejoras en la compatibilidad entre idiomas
  - Modo noche
  - Bloqueo numérico más inteligente que en versiones anteriores
  - Modo VR (Virtualidad Real)
  - Reinicio inteligente
  - Mejoras en velocidad y fluidez
  - Grabación en Android TV

2.1.4. Android frente a otros sistemas operativos

Antes hemos mencionado que el sistema operativo Android está presente en más del 80 % de los dispositivos pero ¿qué ocurre con el 20 % restantes? Existen alternativas como iOS, Ubuntu Mobile, BlackBerry OS o Windows Phone que también están presentes en el mercado y que presentaremos a continuación.

- **iOS**

  Sistema operativo propiedad de Apple orientado a sus dispositivos móviles: iPhone, iPod e iPad. Cuenta con actualizaciones peródicas que

[4]Una versión beta es una versión de pruebas, no completamente estable ni libre de errores.

están disponibles para su descarga y actualización mediante el software iTunes, que es el software gratuito e indespensable para manipular y sincronizar toda clase de archivos en estos dispositivos. Además, hay que destacar que la instalación de iOS no es viable en hardware de terceros.

- **Ubuntu Mobile (Ubuntu Touch)**

  Sistema operativo de código abierto creado por Canonical para instalar Linux en nuestro dispositivo móvil. Actualmente es una versión en pruebas y orientada a fabricantes y compañías telefónicas aunque existe la posibilidad de que un usuario común pueda instalarlo. Los dispositivos soportados son limitados y el lanzamiento de la primera versión incluye una cantidad limitada de aplicaciones básicas como navegador, reloj, calculadora, indicador del tiempo, etc siendo el objetivo de este sistema operativo acercar la interfaz de escritorio a los dispositivos móviles.

- **Blackberry OS**

  Sistema operativo de código cerrado desarrollado por BlackBerry para sus dispositivos. El sistema permite la multitarea y tiene soporte para diferentes métodos de entrada adoptados para su uso en dispositivos móviles. Acceso a correo electrónico, navegación web, sincronización con programas como Microsoft Exchange o Lotus Notes además de las tareas usuales son algunas de las funciones que nos brinda este sistema operativo.

- **Windows Phone**

  Sucesor de Windows Mobile, Windows Phone es un sistema operativo móvil desarrollado por Microsoft pensado para el mercado de consumo. Microsoft ofrece una nueva interfaz de usuario que integra varios de sus servicios como OneDrive, Skype o Xbox Live. Actualmente Microsoft se encuentra trabajando en Windows 10 Mobile, la versión que reemplazará este sistema operativo.

Comparándo las especificaciones de Android frente a estos sistemas operativos podemos extraer las siguientes conclusiones:

- **El código de Android es abierto** (open source) con lo que cualquier persona puede desarrollar una aplicación o modificar el código base posibilitando la adaptación de Android a multitud de dispositivos. Además, gracias a las mejoras de código es posible sacar nuevas versiones sin depender de los fabricantes o las operadoras.

- **No se impone limitación en cuanto a qué se puede instalar** y qué no; es posible instalar cualquier tipo de aplicación, ya sea software de terceros o aplicaciones que nosotros mismos hemos desarrollado.
- **Fomenta la retroalimentación**: Android cuenta con la comunidad más grande de desarrolladores y el mayor movimiento de estos con eventos, concursos, competiciones, etc fomentando de esta forma la participación y la colaboración para encontrar mejoras e ideas para futuras versiones.
- **Android es completamente personalizable.**
- El **sistema multitarea** que incluye es capaz de gestionar varias aplicaciones abiertas a la vez dejando en suspensión aquellas que no se utilicen para evitar un consumo excesivo de memoria.

También tenemos que mencionar que existen una serie de **desventajas** que hacen que determinados usuarios se decanten por algunas de las alternativas que antes hemos mencionado.

- La posibilidad de instalar este SO[5] en cualquier dispositivo hace que además de perder la exclusividad que tendríamos con algunos de sus rivales, su **rendimiento no esté optimizado** para todos los terminales pues como sabemos actualmente hay multitud de dispositivos. Pensemos en iOS, la ventaja de estar presente solo en los dispositivos de Apple hace que el sistema operativo esté totalmente optmitizado para estos terminales.
- Para un principiante, **su uso no es nada intuitivo**.
- **Las actualizaciones están vinculadas al fabricante**, de tal forma que a pesar de que se saque una nueva versión, pasa un tiempo hasta que pueda llegar a nuestros dispositivos pues primero tiene que pasar por manos del operador para hacer las adaptaciones pertinentes (actualizaciones oficiales).
- **Mala gestión de la batería**. Es necesario que el usuario optimice su dispositivo si quiere sacar el máximo partido al rendimiento de su batería.

[5]SO: Sistema operativo

2.2. Php (Hypertext Preprocessor)

Lenguaje de programación de código abierto adecuado para el desarrollo web dada la posibilidad de incorporarlo directamente en un documento HTML en el lado del servidor en vez de tener que llamar a un archivo externo que procese los datos. Así pues, un servidor con un módulo de procesado PHP [2] interpreta este código generando la página web resultante.

Algunas de las características más interesantes de este lenguaje se presentan a continuación:

- Lenguaje fácil de aprender
- El código escrito en php es invisible al navegdor web y al cliente ya que es el servidor quien lo ejecuta
- Capacidad de conexión con la mayoría de motores de bases de datos
- Amplia documentación sobre cómo usarlo
- Lenguaje de programación de código abierto
- Desde Php5 se incluye manejador de excepciones
- Gran flexibilidad a la hora de usarlo
- Se pueden aplicar técnicas de programación orientada a objetos
- Mediante extensiones es posible crear aplicaciones con interfaz gráfica para el usuario

Por todo esto, php suele ser desplegado en la mayoría de los servidores web de la actualidad, estando instalado en más de 20 millones de sitios web y en más de un millón de servidores. Además, el parecido con otros lenguajes de programación como C y Perl permite que la mayoría de programadores puedan explotar su potencial con una curva de aprendizaje muy corta.

2.3. JSON (Javascript Object Notation)

JSON describe los datos con una sintaxis dedicada que se usa para identificar y gestionar los datos permitiendo definirse como un formato usado para el intercambio de datos. Siendo una notación de objetos basada en JavaScript utiliza una sintaxis que nos permite crear objetos de manera rápida y sencilla.

Dado que JSON puede ser leído por cualquier lenguaje de programación es comúnmente usado para el intercambio de información entre distintas tecnologías.

Define los siguientes tipos de valores:

- **Entero o float** (número)
- **String** (cadena de texto)
- **Booleano** (true o false)
- **Array** (colección de datos)
- **Objeto**
- **null**

Para escribir en JSON [3] debemos tener en cuenta lo siguiente:

- Los datos estan separados por comas
- Los datos se escriben en pares (nombre-valor)
- Los objetos se definen entre llaves
- Para definir arrays se utilizan los corchetes

Así pues, un ejemplo de un objeto definido en JSON sería:

```
var objetoJSON = {"Fruta":"sandia", "Azúcares":"25"}
```

2.4. Tecnologías de Red Inalámbrica

Las tecnologías de interconexión inalámbrica van desde redes de voz y datos, que permiten a los usuarios establecer conexiones inalámbricas a través de largas distancias, hasta las tecnologías de luz infrarroja y radiofrecuencia que están optimizadas para conexiones de cortas distancias.

Gracias a este tipo de redes un usuario puede mantenerse conectado mientras se desplaza dentro de una determinada área geográfica.

La frecuencia de transmisión, el alcance o la velocidad de transmisión son algunos de los factores que determinan las diferentes tecnologías que existen.

La instalación de este tipo de redes no requiere ningún cambio significativo en la infraestructura existente como pasa con las redes cableadas ya que la

comunicación se hace mediante ondas electromagnéticas. Esto implica que sea necesaria una regulación que defina rangos de frecuencia y potencias de tranmisión para que el espectro sea aprovechado de una forma eficiente pues, como sabemos, es un bien limitado.

Detrás de esta tecnología existen organizaciones como el IEEE (*Institute of Electrical and Electronics Engineers*), IETF (*Grupo de trabajo de ingeniería de Internet*), WECA (*Wireless Ethernet Compatibility Alliance*), UIT (*Unión Internacional de Telecomunicaciones*) que trabajan continuamente en proyectos de estandarización para tratar de asegurar la interoperabilidad, promover la adopción de tecnologías inalámbricas y reducir costos.

En función de la distancia a través de la que se transmiten los datos existen diferentes tipos de redes inalámbricas que mencionamos a continuación:

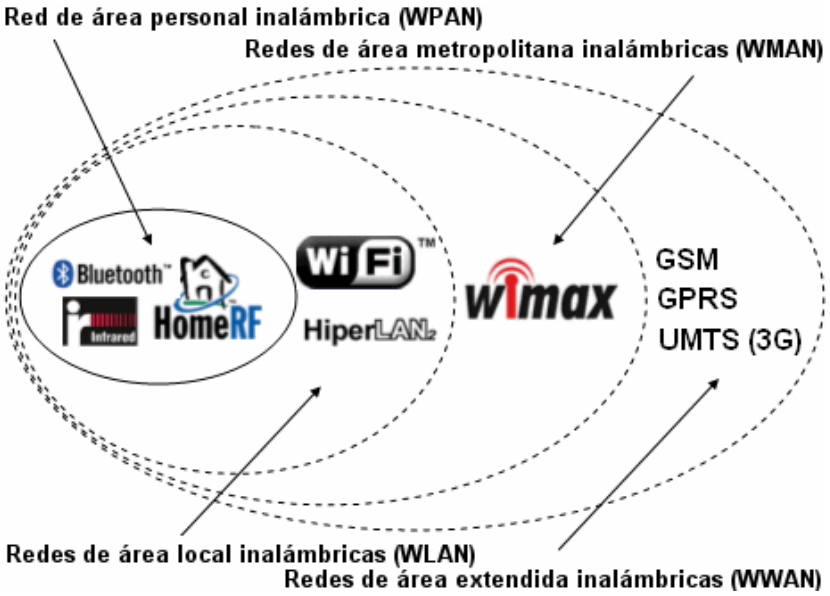

Figura 2.3: Redes inalámbricas

2.4.1. WPAN (Red de área personal inalámbrica)

Una red inalámbrica de área personal incluye aquellas redes de corto alcance que abarcan áreas de algunas decenas de metros. Normalmente se utiliza para conectar periféricos como impresoras, dispositivos móviles o electrodomésticos.

La tecnología principal es **Bluetooth** *(IEEE 802.15.1)* que, lanzado por Ericsson en 1994 ofrece una velocidad máxima de 1 Mbps con un alcance máximo de unos treinta metros.

Existen alternativas como **HomeRF** (*Home Radio Frequency*) lanzada en 1998 por HomeRF Working Group (Incluye compañías como Compaq, HP, Intel, Siemens, Motorola y Microsoft) que ofrece velocidades máximas de 10 Mbps con un alcance de 50 a 100 metros sin amplificador, **Zigbee** (*IEEE 802.15.4*) utilizada para conectar dispositivos a un coste energético y económico muy bajo con una velocidad de transferencia máxima de 250 Kbps a unos 100 metros de alcance o las conexiones **infrarrojas** que se utilizan en conexiones cuya distancia es reducida (pocos metros) alcanzando velocidades de pocos megabits por segundo.

2.4.2. WLAN (Red de área local inalámbrica)

Red que cubre áreas de unos cien metros (aproximadamente el área equivalente a la red local de una empresa). Permite que los dispositivos que se encuentren dentro de ese rango puedan conectarse y comunicarse entre sí. Existen dos tipos de tencologías: Wifi e hiperLAN2.

HiperLAN2 *(High Performance Radio LAN 2.0)* es un estándar europeo desarrollado por ETSI *(European Telecommunications Standards Institute)* que permite a los usuarios alcanzar velocidades de 54 Mbps en área de unos cien metros transmitiendo dentro del rango de frecuencias de 5150 y 5300 MHz.

Por otro lado tenemos la tecnología **Wifi** *(IEEE 802.11)* [4], uno de los focos de este proyecto.

802.11 define el uso de los dos niveles inferiores de la arquitectura OSI: la capa física y la capa enlace de datos. Wi-Fi es el nombre de la certificación otorgada por la Wi-Fi Alliance (anteriormente WECA) que garantiza la compatabilidad entre equipos que utilizan el estándar 802.11. Existen diferentes estándares Wi-Fi que ofrecen distintas velocidades, rangos de frecuencias o aspectos de seguridad (802.11a , 802.11b, 802.11n, etc).

Existen una serie de conceptos que necesitamos conocer si queremos entender su funcionamiento:

- **Estación:** Dispositivo con interfaz de red
- **Medio:** Radiofrecuencia o infrarrojos.
- **Punto de acceso (AP):** Elemento que conecta dos redes con niveles de enlace parecidos o distintos y realiza por tanto las conversiones de tramas correspondientes.
- **Sistema de distribución:** Proporcionan movilidad entre AP.

- **Conjunto de Servicio Básico (BSS):** Grupo de estaciones que se intercomunican entre sí. Cuando las estaciones se intercomunican directamente se denominan independientes y si lo hacen mediante un AP se denomina infraestructura.
- **Conjunto de Servicio Extendido (ESS):** Unión de varios BSS.
- **Área de servicio básico:** Área en la que los dispositivos pueden cambiar de ubicación.

Por último vamos a ver los parámetros que definen una red inalámbrica que utilice este estándar:

- **SSID** (Service Set Identifier)

  Nombre que se muestra públicamente de la señal inalámbrica, es el nombre que aparece cuando nos vamos a conectar a una red Wi-Fi (es posible no mostrar este parámetro si queremos ocultar nuestra red).

- **BSSID** (Basic Service Set Identifier)

  Nombre de identificación único de todos los paquetes de una red inalámbrica para identificarlos como parte de dicha red.

- **RSSI** (Received Signal Strength Indicator)

  El Indicador de fuerza de la señal recibida es una escala de referencia en relación a 1 mW cuya función es medir el nivel de potencia de las señales de redes inalámbricas recibidas o detectadas por un dispositivo.

- **Canal**

  Toda red Wi-Fi transmite y recibe datos en un determinada frecuencia o canal. Como los datos Wi-Fi son digitales, diferentes dispositivos pueden comunicarse usando el mismo canal.

- **Frecuencia**

  Representa la velocidad a la que los datos se transmiten y se reciben entre los dispositivos de una red inalalámbrica.

- **Encriptación**

  Tipo de algortimo que se sigue para encriptar (codificar) los datos que se mandan en una red evitando así que un tercero pueda entender lo que se está transmitiendo. Existen diferentes tipos de encriptaciones:

  - **WEP** (Wired Equivalent Privacy)
    Protocolo de seguridad que proporciona un cifrado a nivel 2, basado en el algortmo de cifrado RC4 que utiliza claves de 64 o

128 bits. Es facilmente vulnerable y por este motivo no es muy utilizado en la actualidad.

- **WPA** (Wi-Fi Protected Access)
  Se utiliza un vector de inicialización de 48 bits y una clave de cifrado de 128 bits. WPA utiliza TKIP [6] para cambiar la clave de cifrado cada vez que un paquete se transmite.
- **WPA2** (Wi-Fi Protected Access 2)
  Versión certificada del estándar de la IEEE de WPA con algunas actualizaciones como el uso de cifrado AES[7].

2.4.3. WMAN (Red de área metropolitana inalámbrica)

También conocida como bucle local inalámbrico *(WLL, Wireless Local Lop)* se basa en el estándar IEEE 802.16 y ofrece una velocidad máxima de 10 Mbps con un alcance de unos 10 km. Actualmente, la mejor red inalámbrica de área metropolitana es **WiMAX** que alcanza la velocidad de 70 Mbps en un radio de varios kilómetros.

2.4.4. WWAN (Red de área extensa inalalámbrica)

Tienen el alcance más amplio de todas las anteriores. Los teléfonos móviles están conectados a este tipo de redes siendo las tecnologías principales: **GSM** *Global System for Mobile Communication*, **GPRS** *General Packet Radio Service* y **UMTS** *Universal Mobile Telecommunication System*.

[6] **TKIP (Temporal Key Integrity Protocol):** Solución que resuleve el problema de reutilización de los Vectores de Inicialización del cifrado WEP. El proceso de TKIP comienza con una clave temporal de 128 bits que es compartida entre los clientes y los puntos de acceso. Combina la clave temporal con la dirección MAC del cliente, luego agrega un vector de inicialización de 16 octetos. Además utiliza RC4 para realizar el cifrado.

[7] **AES(Advanced Encryption Standard):** Uno de los algoritmos de cifrados simétricos más seguros y utilizados hoy en día. No es vulnerable al criptoanálisis diferencial o lineal. Su cifrado se basa en matrices de estado teniendo tamaños de 128, 192 y 256 bits de largo utilizando puertas XOR.

2.5. Software empleado

2.5.1. AndroidStudio

Figura 2.4: Android Studio

Entorno de desarrollo integrado (IDE) basado en IntelliJ IDEA de la compañía JetBrains que proporciona mejoras con respecto al plugin ADT (Android Developer Tools) para Eclipse. Es un software libre programado en Java que utiliza una licencia Apache 2.0 y disponible para Mac OS X, Windows y GNU/Linux [5].

Actualmente se ha presentado como el sustituto oficial de Eclipse para el desarrollo de aplicaciones Android gracias a características como:

- Soporte para programar aplicacioens para Android Wear (SO para dispositivos corporales)
- Herramientas Lint para detectar problemas de rendimiento, usabilidad y compatibilidad entre versiones
- Utilizar ProGuard para optimizar y reducir el código del proyecto al exportar la aplicación
- Integración de Gradle para gestionar y automatizar la costrucción de proyectos
- Interfaz específica para el desarrollo de aplicaciones Android
- Vista previa de proyecto

- Editor de diseño que muestra vista previa de los cambios realizados
- Construcción y gestión de proyectos basado en Maven
- Generador de assets

El uso de este software requiere un equipo que conste con, al menos, las siguientes especificaciones:

- 2GB de memoria RAM
- 400 MB de espacio libre en disco
- 1GB para Android SDK
- Monitor de 1280x800
- Java Development Kit 7 o superior
- GNU Library C 2.15 o superior en el caso de su uso en GNU/Linux

2.5.2. MongoDB

Figura 2.5: MongoDB

MongoDB [6] es una base de datos orientada documentos, lo que quiere decir que en vez de guardar los datos en registros, se guardan en documentos BSON, que es una representación binaria de JSON[8]. Una de as diferencias que existen con respecto a las bases de datos relacionales es que no es necesario

[8]**Json (JavaScript Object Notation)** es un formato para el intercambios de datos, básicamente JSON describe los datos con una sintaxis dedicada que se usa para identificar y gestionar los datos. JSON nació como una alternativa a XML, el fácil uso en javascript ha generado un gran numero de seguidores de esta alternativa. Una de las mayores ventajas que tiene el uso de JSON es que puede ser leído por cualquier lenguaje de programación.

seguir un esquema. Así pues los documentos de una misma colección pueden tener esquemas diferentes.

A pesar de que en MongoDB las consultas se hacen pasando objetos JSON como parámetro, está escrito en C++.

Esta base de datos incluye una consola construida sobre JavaScript desde la que se pueden ejecuar diferentes comandos para utilizar funciones tanto de MongoDB como de JavaScript.

Dadas las ventajas que ofrece sobre las bases de datos relacionales, MongoDB es especialmente útil en entornos en los que se requiera escalabilidad. Gracias a sus opciones de replicación y sharding[9] podemos conseguir un sistema que escale horizontalmente sin problemas.

Por otro lado, si necesitamos garantizar transicciones, MongoDB no es nuestra base de datos ya que solo garantiza operaciones atómicas a nivel de documento.

[9]El **Sharding** es una técnica que consiste en particionar los datos de una base de datos horizontalmente agrupándolos de algún modo que tenga sentido y que permita un direccionamiento más rápido.

Capítulo 3

Planteamiento de la Arquitectura del Sistema

Como se ha indicado en la introducción del presente documento, el objetivo de este proyecto es desarrollar una aplicación que sea capaz de obtener la información disponible sobre las redes inalámbricas que se encuentran disponibles alrededor de nuestro dispositivo móvil y enviarla a un servidor que pueda almacenar esta información para su posterior estudio.

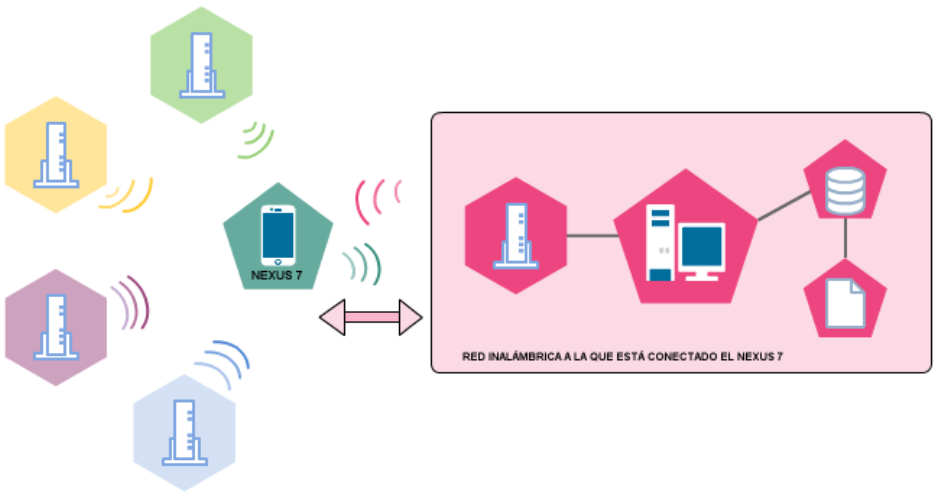

Figura 3.1: Escenario

Así pues podemos distinguir tres elementos en nuestro escenario: El dispositivo móvil donde se encuentra ejecutando nuestra aplicación, la infraestructura a la que estamos conectados y el servidor web al cual mandamos dicha información.

3.1. Dispositivo móvil

A pesar de que la aplicación se desarrollará para que funcione en cualquier dispositivo Android, dos han sido los dispositivos que se han utilizado para el testing de la aplicación desarrollada:

NEXUS 5 16GB

- Android 6.0 (Marshmallow)
- Kernel stock
- Privilegios de superusuario (ROOT)
- Wi-Fi 802.11 a/b/g/n/ac
- Pantalla de 4.95 pulgadas (1080 x 1920 pixels)
- Sensor acelerómetro para auto rotación
- 2 GB de memoria RAM
- CPU: Qualcomm Snapdragon 800, 2,26 GHz
- GPU: Adreno 330 de 450 MHz

Figura 3.2: Nexus 5.

NEXUS 7 32GB + LTE

- Android 6.0 (Marshmallow)
- Kernel stock
- Sin privilegios de superusuario
- Wi-Fi de doble banda (2,4G/5G) 802.11 a/b/g/n
- Pantalla de 7.02 pulgadas (1920 x 1200 pixels)
- Sensor acelerómetro para auto rotación
- 2 GB de memoria RAM

Figura 3.3: Nexus 7.

- CPU: Qualcomm Snapdragon S4 Pro, 1,5 GHz
- GPU: Adreno 320, 400 MHz

El motivo por el cual se han usado varios dispositivos a la hora de probar nuestro software ha sido para demostrar la funcionalidad de nuestra aplicación en todo tipo de dispositivo independientemente del tamaño de pantalla o especificaciones.

3.2. Infraestructura

Alrededor de nuestro dispositivo existen multitud de redes inalámbricas disponibles a las que podríamos conectarnos en caso de tener los correspondientes datos de acceso. De cada una de estas redes nuestra aplicación recibirá datos como la fuerza de la señal con la que el dispositivo detecta la red, el canal o por ejemplo la frecuencia en la que se está transmitiendo la señal. Toda esta información será analizada para detectar la red que es más conveniente para nosotros pudiendo enviar toda esta información a un servidor dado.

Ahora bien, para enviar esta información al servidor, se ha creado una red inalámbrica a la que nuestro dispositivo Nexus 7 estará conectado y que usará para enviar la información recogida. Ha sido necesario que tanto el dispositivo como el servidor estén en la misma red por el hecho de que el servidor es local, es decir, estamos utilizando nuestro ordenador como servidor local y no un servicio en la nube para las pruebas del software.

Por ello ha sido necesario crear una red inalámbrica utilizando la red a la que estaba conectado nuestro ordenador[1]. Es decir, lo que hemos hecho ha sido configurar nuestro ordenador como punto de acceso (Ad-hoc) donde nuestro dispositivo móvil sea capaz de conectarse.

A continuación mostraremos cómo conseguimos configurar nuestro ordenador como punto de acceso inalámbrico utilizando nuestra conexión de área local. Indicar que esta tarea fue sencilla gracias a estar utilizando equipos que corrían bajo un sistema operativo basado en Linux.

- Instalamos el software hostapd
- Abrimos en modo edición el fichero /etc/hostapd/hostapd.conf

[1]Esta fue la única posibilidad que existía dadas las políticas internas de seguridad de la empresa para la que se realizó el proyecto presentado.

- Establecemos los parámetros deseados[2] de nuestro punto inalámbrico, en nuestro caso:

```
interface=wlan0
driver=nl80211
ssid=EXAMPLEWLAN
hw_mode=g
channel=1
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=3
wpa_passphrase=pass1234
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

- Indicamos a hostapd que debe utilizar el fichero anteriormente modificando el fichero /etc/default/hostapd

```
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

- Iniciamos el hostapd
  - $ sudo service hostapd start
- Desde nuestro dispositivo móvil detectamos la red y usamos la configuración manual para darle una IP perteneciente a la misma subred a la que pertenece el ordeandor.
- Paramos el servicio cuando queramos dejar de usarlo
  - $ sudo service hostapd stop

3.3. Servidor web

Como hemos indicado anteriormente, utilizaremos nuestro ordenador (el mismo que hemos configurado como punto de acceso) como servidor local. En este instalaremos MongoDB como base de datos para almacenar los datos que nos envie nuestro dispositivo almacenando la información en forma de documentos para su posterior análisis.

Existen varios motivos por los que hemos seleccionado una base de datos no relacional: En primer lugar por la velocidad a la hora de consultar los

[2]Esta ha sido la configuración para nuestro caso de estudio, si queremos una configuración diferente podemos obtener más información en: http://linuxwireless.org/en/users/Documentation/hostapd/

datos, por la escalabilidad que nos ofrece, por no obligarnos a seguir un esquema fijo a la hora de guardar los datos (la estructura que seguiremos podemos definirla nosotros mismos gracias a JSON) y por guardar las tablas como documentos. Pensemos que guardar los datos recogidos por nuestro dispositivo en una tabla convencional no sería eficiente ni escalable cuando la cantidad de información sea muy elevada.

Adelantamos que para que nuestro servidor (base de datos) pueda entender la información enviada desde el dispositivo necesitaremos disponer de un script Php que esté escuchando peticiones y tener los datos enviados en formato JSON. Entraremos más en detalle en el siguiente capítulo.

Capítulo 4

Desarrollo de la Aplicación: InfoWifiApp

En este capítulo explicamos al lector todo lo relacionado con la aplicación desarrollada: Los requisitos necesarios para su funcionamiento, la interfaz de usuario y las funcionalidades implementadas, la arquitectura interna, el proceso de intercambio de información con el servidor correspondiente y los análisis de rendimiento de la aplicación.

4.1. Requisitos

InfoWifiApp ha sido diseñada para funcionar en dispositivos que corran bajo Android 4.4 (KitKat)[1] o superior y que consten de una tarjeta inalámbrica que permita detectar y conectarse a redes inalámbricas.

Figura 4.1: InfoWifiApp.

Pero a parte de estas dos condiciones necesarias, existen una serie de permisos que tenemos que aprobar para que la aplicación pueda obtener información sobre las redes inalámbricas que existen a nuestro alrededor:

- Access Wifi State
- Change Wifi State

[1]El motivo por el que se ha puesto esta versión como mínima es debido a que es a partir de esta versión cuando podemos utilizar las librerías correspondientes para la gestión de memoria en aplicaciones que corran en segundo plano.

- Write External Storage
- Access Network State
- Internet Access

Una vez aceptados, nuestra aplicación podrá utilizar el módulo Wifi, podrá escribir en ficheros internos y tendrá la posibilidad de conectarse a internet.

En caso de no tener el Wifi activado a la hora de usar la aplicación esta mostrará un mensaje de error indicando que se ha de activar dicho módulo para poder utilizar la aplicación.

4.2. Inferfaz de usuario

La interfaz de usuario ha sido diseñada de tal forma que es posible utilizar el dispositivo tanto en horizontal como en vertical.

Nada más abrir la aplicación nos encontraremos con cuatro pestañas totalmente accesibles por el usuario:

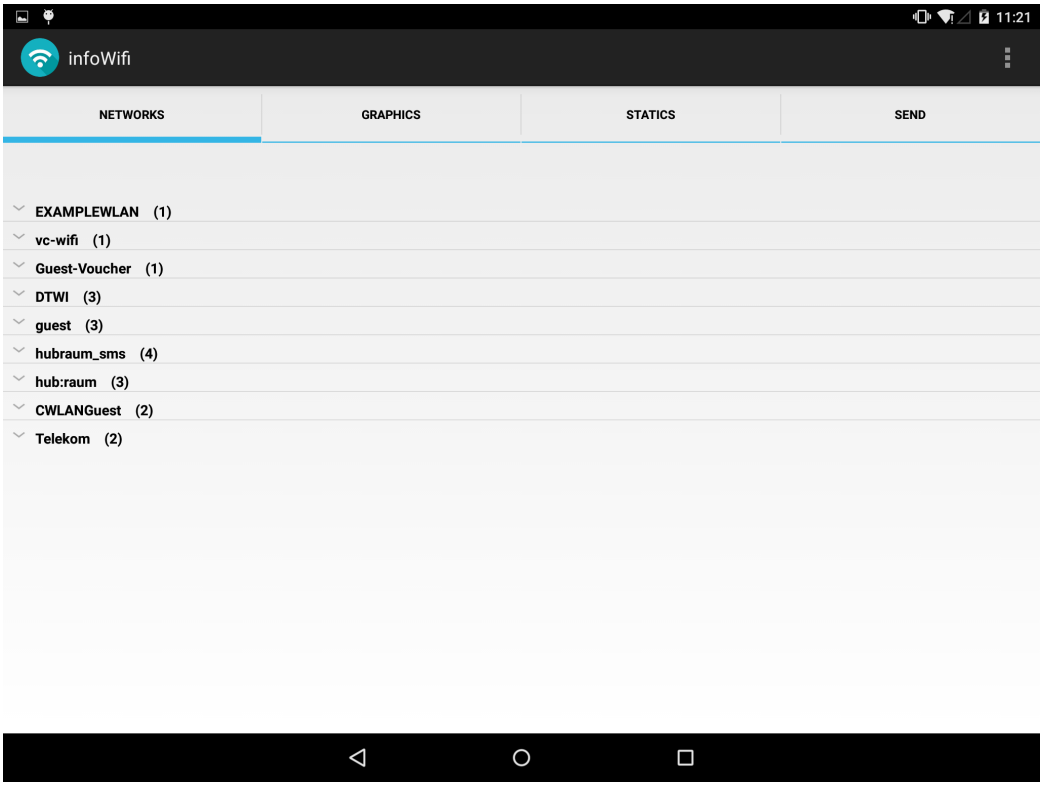

Figura 4.2: Pantalla principal

- **Networks**

  Por defecto, la pestaña Networks será la primera que se le muestre al usuario cuando este abra la aplicación mostrando las redes inalámbricas disponibles que existen a su alrededor agrupadas por su SSID.

  El motivo por el cual se agrupan las redes por su SSID es porque en muchas ocasiones nos encontraremos en situaciones en las que tengamos varias redes con el mismo SSID pero con distintos parámetros como pueden ser la fuerza de la señal recibida o el BSSID.

  Si el usuario selecciona cualquiera de los SSID mostrados, se desplegarán todas las redes disponibles con ese SSID. Para cada una de las redes se mostrarán los siguientes parámetros: BSSID, Canal, Fuerza de la señal recibida, Frecuencia y encriptación utilizada.

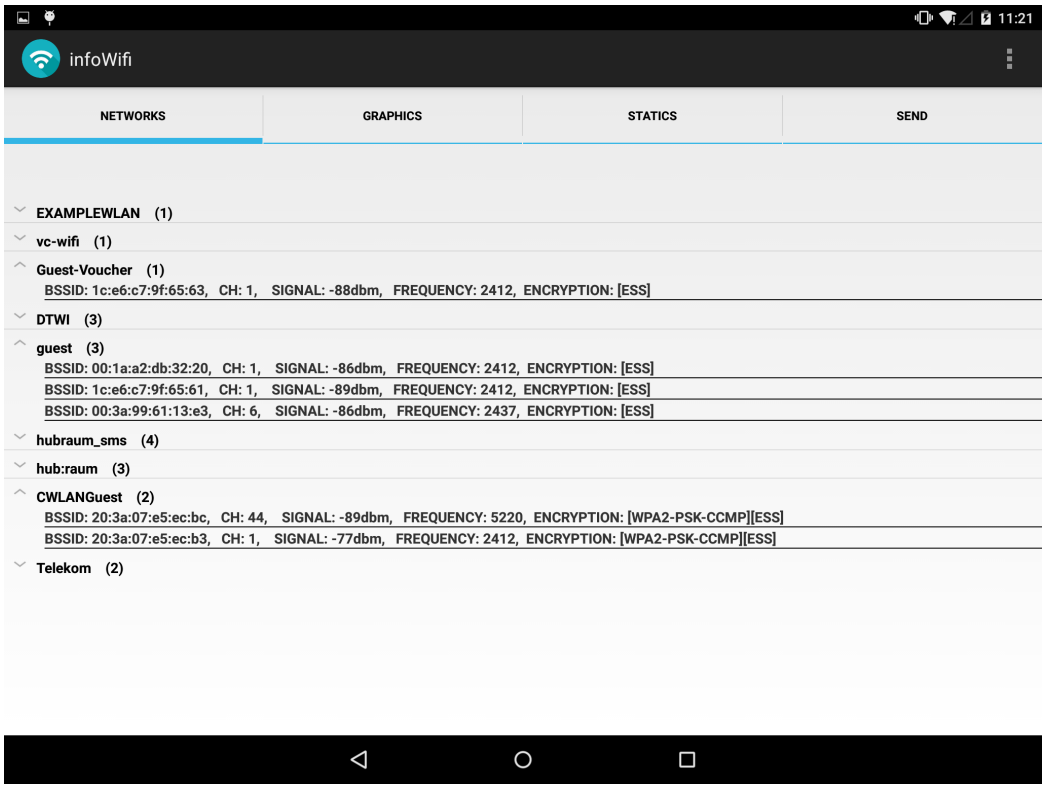

Figura 4.3: Pestaña Networks

La información mostrada en esta pestaña irá siendo actualizada según se produzcan cambios en los parámetros de las redes detectadas. Esto quiere decir que la información mostrada cambiará cada vez que cualquier parámetro cambie o cada vez que estemos en otro punto donde detectemos más o menos redes.

Esto ha sido posible gracias al uso de la clase MainActivity que explicaremos la siguiente sección.

- **Graphics**

  En muchas ocasiones al usuario le gustará comparar las redes disponibles para elegir a cuál conectarse, ver qué canal es el menos usado o ver qué red tiene la señal más fuerte. Para dar respuesta a estas preguntas ha sido implementada la pestaña Graphics, para mostrar al usuario gráficos en los que se muestran todas las redes disponibles y el canal que utilizan (en el caso de pulsar en el botón Channels) o la fuerza de la señal con la que se detecta la red (en el caso de que el usuario pulse el botón Signal).

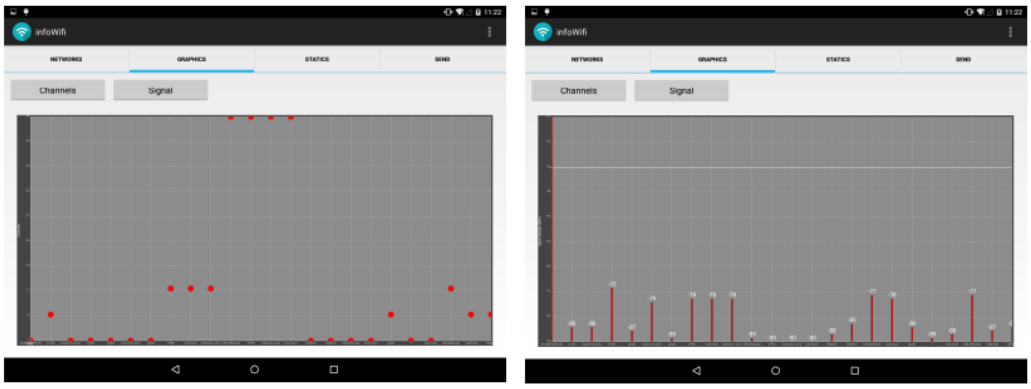

Figura 4.4: Pestaña Channels Graphics

Además se ha dotado al usuario con la posibilidad de manipular estas gráficas de forma fácil y sencilla: ampliar pellizcando la pantalla o desplazarse por los ejes desplazando el dedo por este. Para que esto fuese posible, ha sido necesario el uso de la instancia **OnTouchListener** que explicaremos más adelante.

- **Statistics**

  Como hemos visto hasta este momento, tenemos una gran cantidad de información que agrupándola de la forma adecuada nos da respuestas a preguntas como: ¿Cuál es la mejor red a la que conectarme? ¿Qué canal es el menos usado? ¿Qué redes tienen encriptación WPA2? ¿Cuántas redes hay con el mismo SSID? ¿Qué sé acerca de mi conexión actual?

  Así pues, cuando el usuario pulse el botón statistics tendrá la posibilidad de obtener respuesta a todas estas preguntas pulsando los diferentes botones que se le ofrecen:

  - **Channels**

    Si pulsamos en la pestaña canales veremos todos los canales que no están siendo utilizados por ninguna red inalámbrica Wifi y cuál sería el más adecuado para utilizar. La elección se basa en

un algoritmo con el que se selecciona el canal menos utilizado que interfiere con el menor número de canales adyacentes.

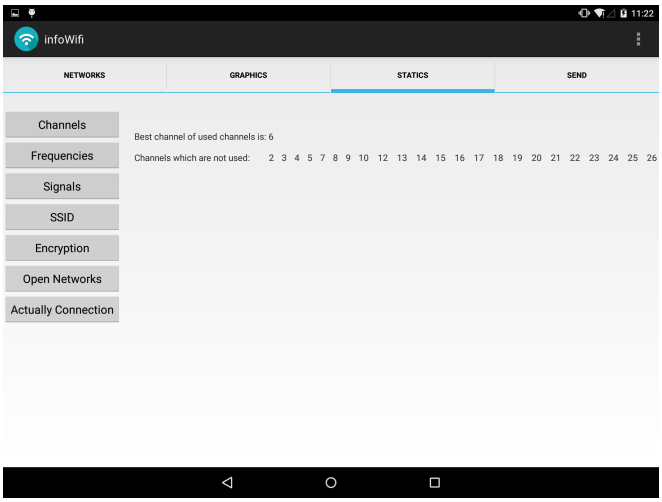

Figura 4.5: Channels Statistics

- **Frequencies**

  Como sabemos, es posible utilizar diferentes frecuencias para transmitir una misma señal: podemos utilizar la banda de 2.4GHz, 5Ghz u otras. Así pues, en este caso podremos ver cuáles de nuestras redes utilizan cada frecuencia.

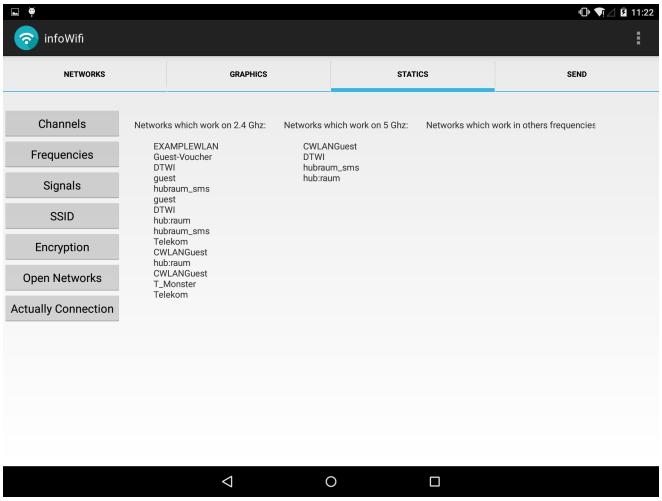

Figura 4.6: Frequencies Statistics

- **Signals**

  Si queremos saber cuál es la mejor red a la que conectarnos en términos de fuerza de señal recibida simplemente debemos pulsar

este botón y podremos ver cuál es la red que detectamos con mayor fuerza.

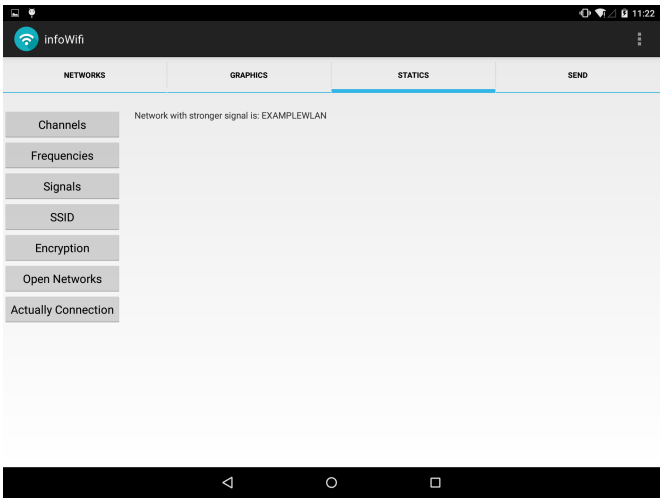

Figura 4.7: Signals Statistics

- **SSID**

  Como dijimos en apartados anteriores, existe la posibilidad de tener varias redes inalámbricas con el mismo SSID. Por este motivo al pulsar este botón veremos tanto gráficamente como textualmente dicha información.

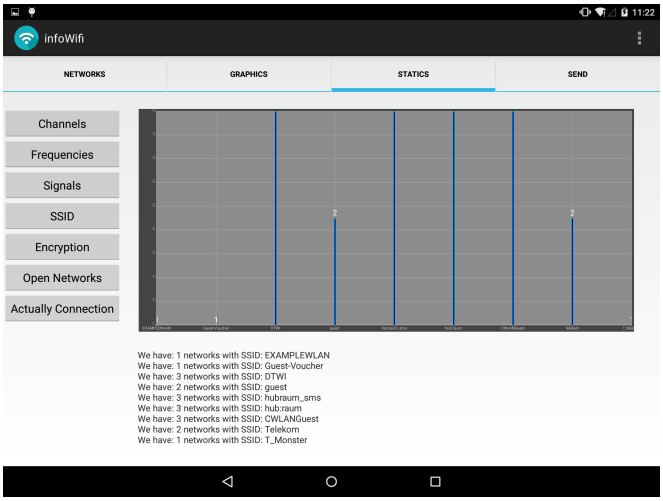

Figura 4.8: SSID Statistics

- **Encryption**

  Quizás este es uno de los apartados más interesantes. En este caso se hace un estudio sobre la encriptación que utilizan las redes inalámbricas detectadas de tal forma que se agrupan las redes con

la misma encriptación y se muestra tanto gráfica como textualmente dicha información para que el usuario pueda ver cuál es la encriptación más utilizada y en qué porcentaje.

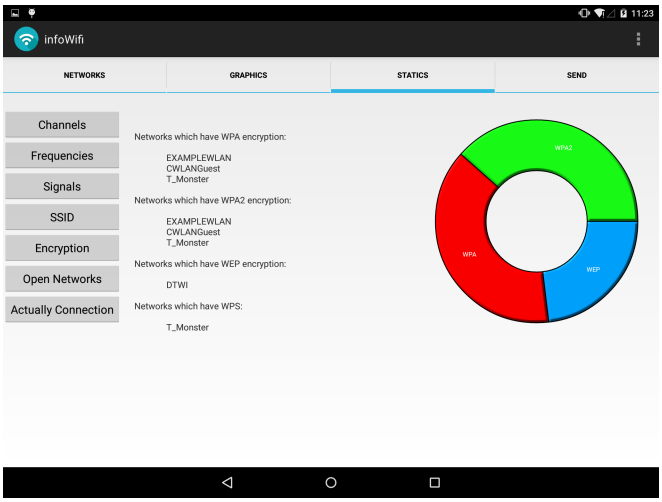

Figura 4.9: Encryption Statistics

- **Open Networks**
  A raíz del análisis anterior es posible detectar a qué redes inalámbricas podríamos conectarnos sin necesidad de tener una clave de acceso. Estas son mostradas en esta pestaña.

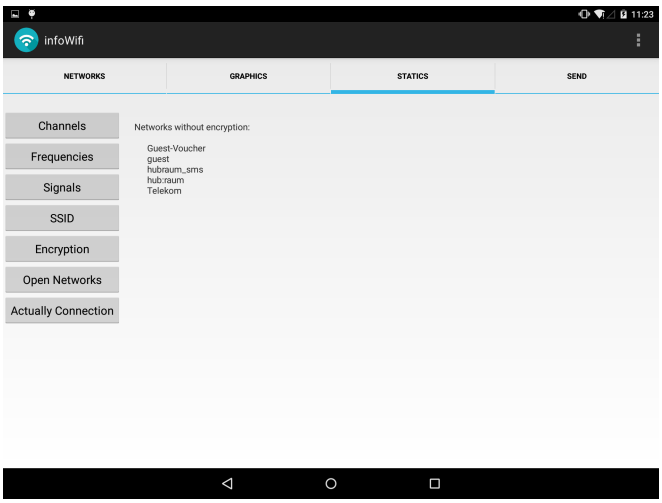

Figura 4.10: Networks without encryption

- **Actually Connection**
  Hasta ahora nos hemos centrado en conocer información sobre las redes que detectamos pero muchas veces nos interesa saber información sobre la red inalámbrica a la que estamos conectados (en

caso de estarlo, claro). Así pues, en esta pestaña nos encontramos con información sobre la red a la que estamos conectados: Cuál es la velocidad del enlace, Cuál es la dirección IP y MAC o el NetworkID entre otros parámetros.

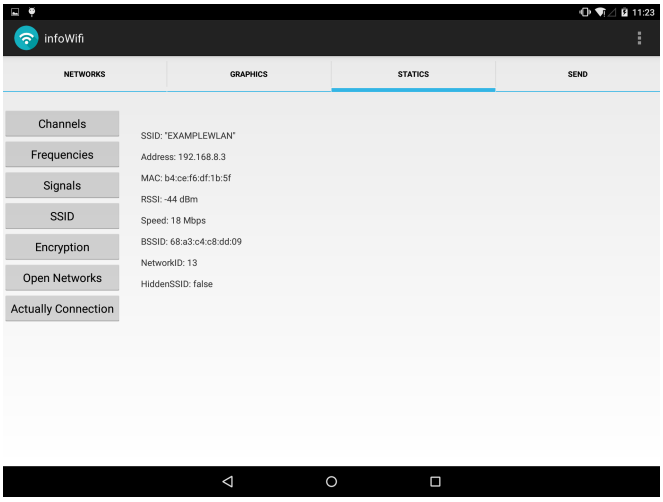

Figura 4.11: Actually Connection

- **Send**

  La pestaña Send ha sido diseñada para sincronizar nuestra aplicación con un servidor que reciba los datos recopilados por nuestro dispositivo. Para ello solo tenemos que escripir la dirección IP en la que se encuentra alojado nuestro servidor, la frecuencia con la que queremos que se envien los datos detectados por nuestro dispositivo e iniciar el proceso pulsando el botón "Start sending!".

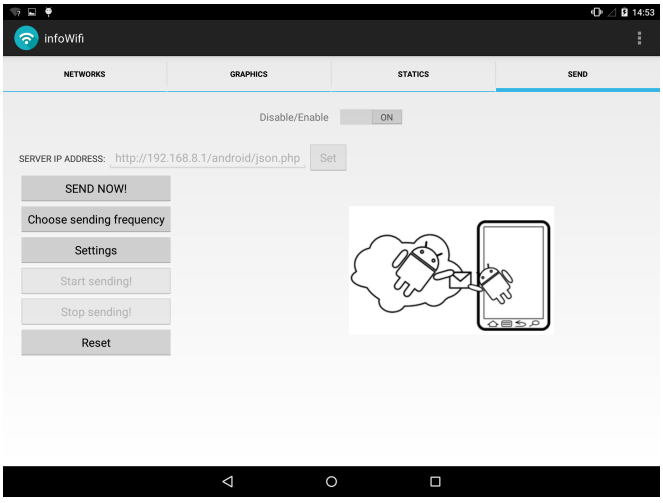

Figura 4.12: Send Tab

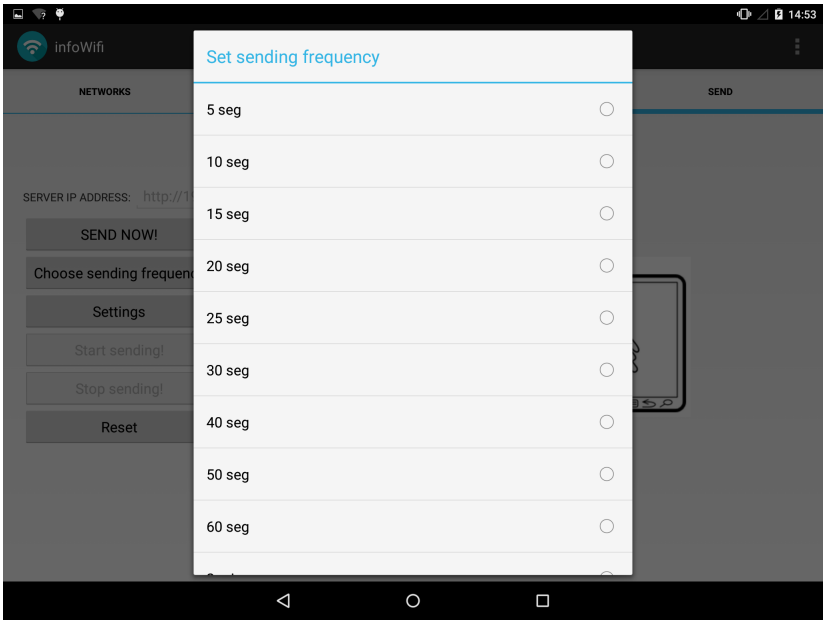

Figura 4.13: Send Tab Settings

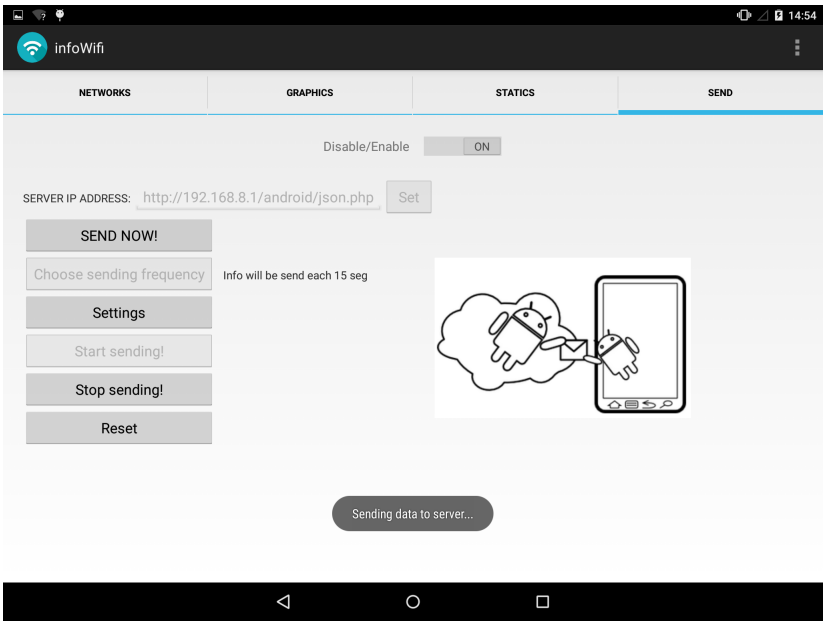

Figura 4.14: Send Tab running

Pero además de estas pestañas, en la esquina superior derecha se ha dotado a la aplicación con un botón adicional que muestra un menú desplegable con dos opciones:

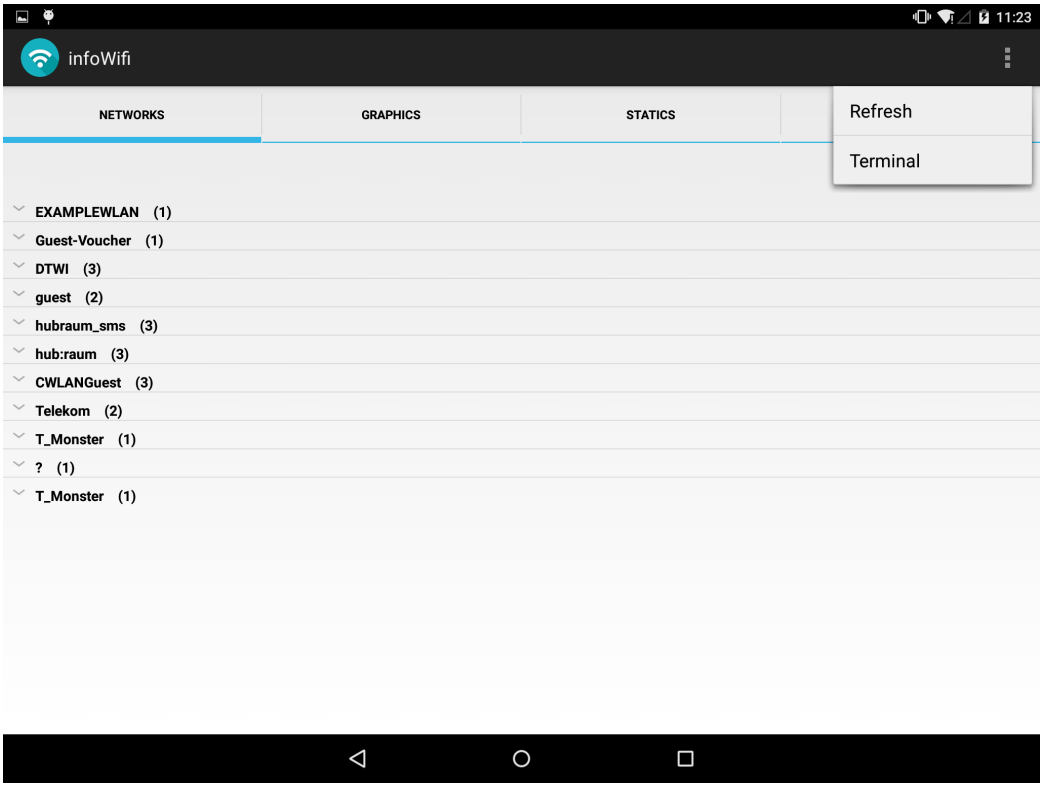

Figura 4.15: More options

- **Refresh**

  Pulsando en Refresh forzamos a que la aplicación actualice los datos detectados volviendo a comprobar las redes existentes sin necesidad de esperar a que se produzca algún cambio en estas.

- **Terminal**

  Sin embargo, si pulsamos el botón Terminal nos encontramos con una funcionalidad extra que se ha añadido a nuestra aplicación: Una consola de comandos.

  Con esta consola es posible utilizar cualquier comando que podamos imaginar, podemos copiar archivos, editar ficheros o ver datos sobre nuestro dispositivo que mediante otros procesos no podriamos ver.

  En caso de tener un dispositivo con superprivilegios, esta consola ofrece un abanico de posibilidades extremadamente superior en comparación con un usuario estándar.

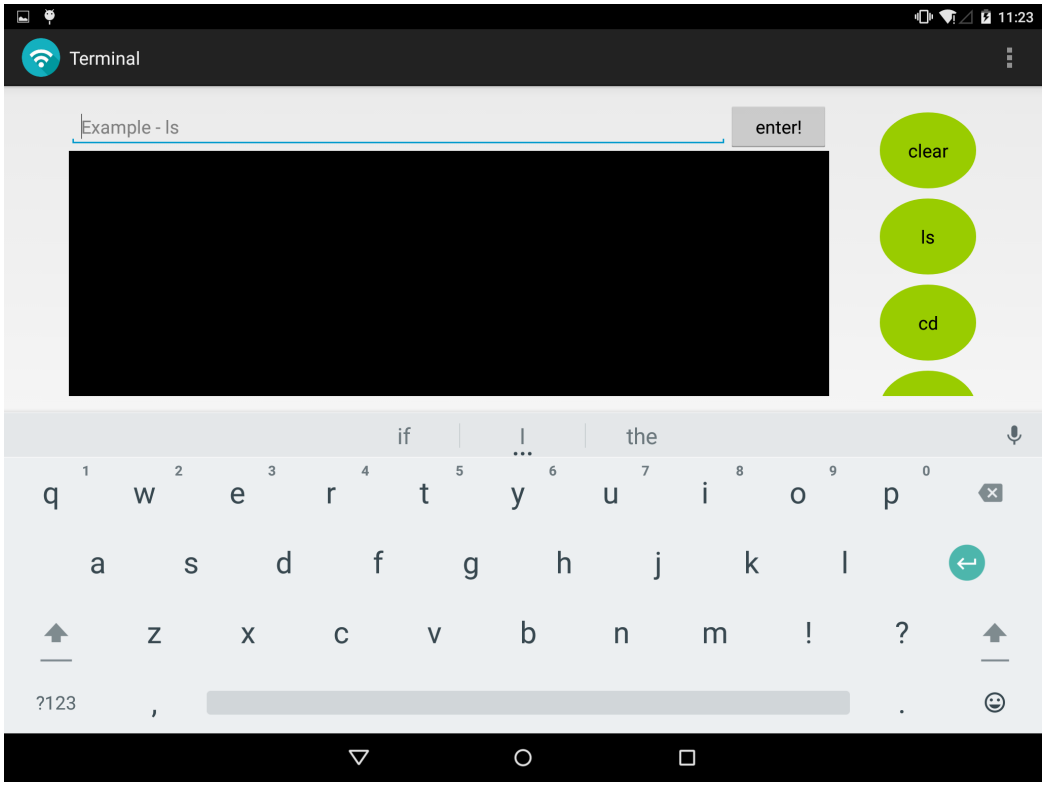

Figura 4.16: Shell

4.3. Arquitectura interna

Toda aplicación que desarrollemos para Android sigue una estructura que se compone del código fuente en sí, archivos de recursos, vistas, librerías de código y el fichero android manifest. Veamos como hemos estructurado nuestra aplicación:

4.3.1. Directorio src

En el directorio src encontramos toda la lógica de nuestra aplicación, es decir, todas las clases programadas. Las diferentes clases que hemos utilizado han sido las siguientes:

MainActivity.java Podemos considerar esta clase como el núcleo de nuestra aplicación. MainActivity.java incluye todo el código necesario para que cada uno de los componentes que usamos en nuestra aplicación puedan funcionar de manera correcta. Se dota a las gráficas de una interfaz táctil, se asocia a cada elemento con la activity adecuada y por supuesto, se implementa el código necesario para que la aplicación pueda funcionar en su condición

más básica: escaneando redes inalámbricas y mostrando la información al usuario.

Adapter.java Para mostrar las redes inalámbricas disponibles a nuestro alrededor utilizamos el elemento ListView que nos permite exponer los datos en una lista agrupable y expandible. Dado que nuestros datos son un tanto especiales (pues son objetos ItemsGroup que hemos creado nosotros mismos), se ha creado esta clase que permite adaptar este tipo de datos a los que se usarían por defecto.

ItemsGroup.java Con esta clase java representamos el conjunto de redes que tengan el mismo SSID.

DoBackground.java Una de las clases más interesantes de nuestra aplicación: DoBackground.java Esta clase es la encargada de hacer posible el envio de datos entre nuestro dispositivo y el correspondiente servidor. Esta clase ha sido diseñada de tal forma que sea posible que la funcionalidad que implementa se pueda estar ejecutando mientras estamos ejecutando otra tarea o estamos fuera de nuestra aplicación (de ahí su nombre).

SendingTime.java Como hemos indicado en apartados anteriores, la información que enviamos al servidor va sellada con el dispositivo que la envió y el tiempo en el que lo hizo. Esta clase es necesaria para crear un objeto de tipo Time que permita a nuestra aplicación obtener la hora en un momento determinado.

Terminal.java Una de las funcionalidades extra con la que hemos dotado a nuestra aplicación ha sido una consola con la que poder interactuar con nuestro dispositivo de una forma más profunda y técnica. Esta clase incluye el código necesario para hacer esto posible.

Commands.java Esta clase a pesar de no ser estrictamente necesaria ha sido creada para futuras mejoras o implementaciones de nuestra aplicación. Básicamente se prepara a la consola con una serie de comandos básicos que pueden ejecutarse de manera cómoda y sencilla sin necesidad de tener que teclearlos.

4.3.2. Directorio lib

En la carpeta lib se incluyen aquellas librerias que son necesarias para dotar a nuestra aplicación con funcionalidades que no se incluyen en las librerias por defecto. Para nuestra aplicación solo ha sido necesario importar la librería **AndroidPlot** para la creación de gráficos dinámicos y estáticos.

4.3.3. Directorio res

Directorio drawable :

Hemos utilizado esta carpeta para almacenar imágenes externas que se muestran en nuestra aplicación (la que aparece en el icono de la aplicación y la que se usa como fondo en la opción de configurar los parámetros para enviar la información al servidor) y para definir aquellos estilos de botones que no estan definidos por defecto (como el botón con forma de óvalo que utilizamos cuyo código se presenta a continuación).

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android
    ">
    <!--Border color-->
    <stroke
        android:width="1dp"
        android:color="@color/blue" />
    <!--Background color-->
    <solid
        android:color="@color/green"/>
    <padding
        android:left="10dp"
        android:top="13dp"
        android:right="10dp"
        android:bottom="13dp" />
</shape>
```

Directorio layout :

A la hora de diseñar la interfaz gráfica de usuario, este es uno de los directorios más importantes ya que en él se definen los distintos layout que se mostrarán al usuario en cada ocasión: cuando pinche en la pestaña enviar, cuando use la función escanear o por ejemplo cuando utilice la consola. Nuestra aplicación ha sido implementada de tal forma que existen un layout para cada función y es desde las clases donde elegimos qué layout mostrar al

usuario. Este criterio se ha seguido así ya que para futuras mejoras o modificaciones podremos fácilmente separar los detalles estéticos de los funcionales.

Directorio xml :

En una de las funcionalidades de nuestra aplicación se muestra al usuario una gráfica de sectores representando qué redes de las analizadas utilizan encriptación Wpa, Wpa2, Wep u otras. En este directorio se incluyen los ficheros xml que nos brindan la posiblidad de definir distintos colores para cada una de las opciones disponibles: por ejemplo el siguiente fichero define el color rojo (#ff0000) para la encriptación wpa.

```
<?xml version="1.0" encoding="utf-8"?>
<config
        fillPaint.color="#ff0000"
        labelPaint.textSize="@dimen/pie_segment_label_font_size"
    />
```

4.3.4. Fichero Android Manifest

El fichero AndroidManifest.xml contiene información esencial necesaria sobre el sistema Android, información que además es necesaria antes de poder ejecutar cualquier línea de código. En este xml se declaran las distintas actividades, se solicitan los permisos de la aplicación, se generan los requisitos del sistema y la SDK requerida y se da un ID a nuestra aplicación.

A continuación mostramos el AndroidManifest que hemos utilizado en nuestra aplicación:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/
    android"
     package="com.example.raul.infowifi" >

    <uses-permission android:name="android.permission.
        CHANGE_WIFI_STATE" />
    <uses-permission android:name="android.permission.
        ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.
        WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.
        ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.INTERNET"
        />

    <application
         android:allowBackup="true"
```

```
        android:icon="@drawable/app"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            android:screenOrientation="landscape" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN
                    " />

                <category android:name="android.intent.category.
                    LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".Terminal"
            android:label="@string/title_activity_terminal"
            android:screenOrientation="landscape" >
            >
        </activity>
        <activity
            android:name=".commands"
            android:theme="@android:style/Theme.Dialog"
            android:label="@string/title_activity_commands" >
        </activity>
    </application>

</manifest>
```

Como podemos ver en primer lugar declaramos los permisos que necesitará nuestra aplicación para poder funcionar adecuadamente. Después declaramos las distintas activities[2] que tendrá nuestra aplicación, que en nuestro caso son tres: MainActivity, Terminal y Commands indicando parámetros como la orientación deseada, la etiqueta que queremos usar o el intent asociado.

[2]Una activity representa una unidad de interacción con el usuario. Toda activity debe tener una vista asociada que será utilizada como interfaz de usuario.

4.4. Intercambio de información

En secciones anteriores se ha comentado la posibilidad de enviar los datos recopilados por nuestro dispositivo a un servidor, veamos como es esto posible:

4.4.1. Escenario planteado

Figura 4.17: Escenario planteado

4.4.2. Procesos llevados a cabo

Recolección de datos :

Tras comprobar que el módulo Wifi se encuentra activado en nuestro dispositivo, utilizamos la clase **WifiManager** que nos proporcionará la API necesaria para administrar aspectos de conectividad Wifi de nuestro dispositivo como:

- La lista de redes configuradas
- En caso de existir, la red que se encuentra activa.
- Los resultados de las exploraciones de puntos de acceso.

Necesitamos saber cuándo se ha detectado una red nueva, cuando la fuerza de las señales cambian o cuando deja de ser viable conectarse a una red;

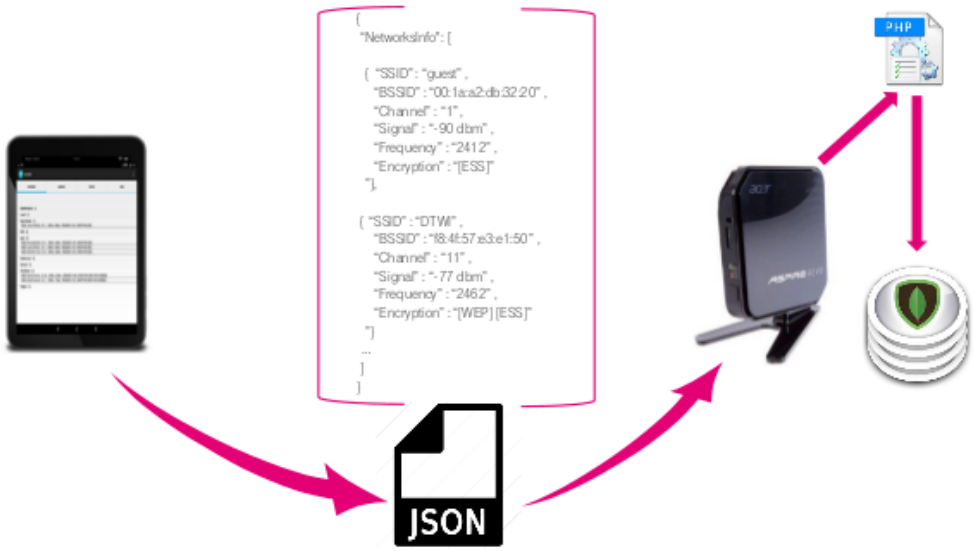

Figura 4.18: Formato de envio de la información

para ello necesitamos un capturador de acciones que notifiquen a nuestra aplicación que algo ha ocurrido. Aquí es donde entra en juego la clase **BroadcastReceiver**, un capturador de acciones que se ejecutará cuando eventos como los indicados anteriormente ocurran.

Teniendo la acción capturada, procederemos a almacenar esta información de tal forma que cada una de las redes escaneadas sea un elemento ArrayList<String>de un objeto tipo List.

Por último se presenta al usuario la información capturada en una lista agrupable por el nombre de la red.

En el código que mostramos a continuación se puede ver como se recuperan los datos, como se agrupan según su SSID y como se muestra al usuario.

```
public final BroadcastReceiver wifireceiver = new
    BroadcastReceiver() {

    public void onReceive(Context context, Intent intent) {

        results = wifimanager.getScanResults();

        for (int i = 0; i < results.size(); i++) {
            if (results.get(i).SSID.equals("") == true) {
                results.get(i).SSID = "?";
            }
        }

        names = new ArrayList<String>();
```

```
int a = 0;
int c = 0;

for (int i = 0; i < results.size(); i++) {

    for (int j = 0; j < names.size(); j++) {
        if (results.get(i).SSID.equals(names.get(j)) ==
            true) {
            a = 1;
            break;
        }
    }

    if (a == 0) {
        names.add(results.get(i).SSID);
        ItemsGroup newItem = new ItemsGroup(results.get(
            i));
        newItem.children.add(results.get(i));

        for (int b = i + 1; b < results.size(); b++) {
            if (results.get(i).SSID.equals(results.get(b
                ).SSID) == true) {
                newItem.children.add(results.get(b));
            }
        }

        groups.append(c++, newItem);
    }

    a = 0;
}

listView = (ExpandableListView) findViewById(R.id.
    listViewexp);
adapter adapter = new adapter(MainActivity.this, groups)
    ;
listView.setAdapter(adapter);
```

Envio de datos :

Como hemos indicado en el apartado anterior, cada una de las redes escaneadas es guardada como un elemento ArrayList<String>de un objeto tipo List.

Pero ahora bien, para poder enviar la información a nuestro servidor y que este la entienda tenemos que encapulsar los datos: usaremos para ello JSON.

```
JSONArray jsonArray = new JSONArray();
JSONObject jsonObj;
```

```
SimpleDateFormat sdf = new SimpleDateFormat("dd'/'MM'/'y␣HH:mm:
    ss");
String currentDateandTime = sdf.format(new Date());

StringBuilder stamp=new StringBuilder();
stamp.append(currentDateandTime);
stamp.append("␣from:␣" + android.os.Build.MODEL + "␣(" + myMac +
    ")");

for(int i=0; i<results.size(); i++){

    jsonObj = new JSONObject();
    jsonObj.put("SSID", results.get(i).SSID);
    jsonObj.put("BSSID", results.get(i).BSSID);
    jsonObj.put("Channel", convertFrequencyToChannel(results.get
        (i).frequency));
                jsonObj.put("Signal", results.get(i).level);
                jsonObj.put("Frequency", results.get(i).
                    frequency);
                jsonObj.put("Encryption", results.get(i).
                    capabilities);
    jsonArray.put(jsonObj);

}

List<NameValuePair> nameValuePairs = new ArrayList<NameValuePair
    >();
nameValuePairs.add(new BasicNameValuePair("req", jsonArray.
    toString()));
nameValuePairs.add(new BasicNameValuePair("TimeStamp", stamp.
    toString() ));

// Use UrlEncodedFormEntity to send in proper format which we
    need
httppost.setEntity(new UrlEncodedFormEntity(nameValuePairs));

// Execute HTTP Post Request
HttpResponse response = httpclient.execute(httppost);
InputStream inputStream = response.getEntity().getContent();
InputStreamToStringExample str = new InputStreamToStringExample
    ();
String responseServer = str.getStringFromInputStream(inputStream
    );
Log.e("response", "response␣-----" + responseServer);
```

Lo que hacemos es convertir cada una de las redes detectadas en un objeto JSON y guardarlo en un array de objetos JSON.

A la hora de enviar la información al servidor se enviará este array de objetos JSON con otro parámetro que indica la hora a la que se envió la información y el dispositivo desde el que se envió.

Una vez tenemos toda nuestra información en un formato legible para el receptor procedemos a realizar el Http Post Request quedándonos a la espera de una respuesta para que el usuario tenga constancia de si el proceso de envío ha sido satisfactorio o no.

Por último también es necesario destacar que este proceso se ha implementado de tal forma que pueda correr en background; lo que quiere decir que el envio de datos se puede hacer mientras estamos haciendo cualquier gestión dentro de nuestra aplicación o incluso fuera de esta.

Recepción de datos :

Para atender la petición HttpPostRequest que hemos enviado desde el dispositivo será necesario tener un pequeño programa escrito en php corriendo en nuestro servidor que atienda dichas peticiones.

El programa que hemos desarrollado para este fin es el expuesto a continuación:

```
<?php

        $username= "android"
        $password= "telekom"

        $req = $_POST['req'];
        $TimeStamp = $_POST['TimeStamp'];

        try {

                $mongo = new MongoClient("mongodb://localhost",
                    array("username" => $username,"password" =>
                    $password));

                $dbname = $mongo->selectDB("AndroidFiles");

                $collection = $dbname -> $TimeStamp;

                foreach($req as $element){
                        $collection -> insert ($element);
                }

                echo "Successful"

        } catch(MongomoException $exception) {
                echo "Something wrong happened";
        }

?>
```

Como se puede apreciar, se recogen de la petición dos parámetros: la marca que indica qué dispositivo envió los datos y la información en sí. Teniendo esto almacenamos los datos en nuestra base de datos controlando las excepciones que puedan surgir y enviando una respuesta al dispositivo para informar si todo ha ido bien o no.

Conectándonos ahora a nuestra base de datos podemos ver como resultado final la siguiente imagen:

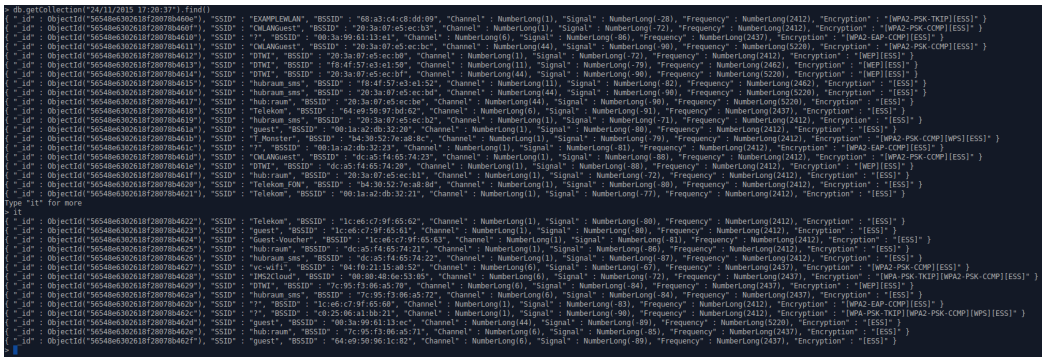

Figura 4.19: Servidor

Almacenamiento de datos :

Como hemos indicado anteriormente, MongoDB es una base de datos orientada a documentos por lo tanto en nuestro servidor encontraremos que los datos enviados por nuestros dispositivos quedan como documentos en formato BSON[3].

Como curiosidad tenemos que destacar que finalmente nos decidimos por utilizar esta base de datos ya que podríamos crear nuestra propia estructura de datos tanto para enviarlos como para almacenarlos permitiendo así tener una comunicación sencilla entre dispositivo y servidor que en caso de haber usado bases de datos relacionales no hubieramos tenido.

[3]**BSON** (Binary JSON) es una serialización codificada en binario de documentos JSON.

4.5. Análisis de rendimiento

Uno de los aspectos más importantes a la hora de desarrollar una aplicación es el consumo de memoria y cpu de esta ya que será un factor determinante para no consumir la batería de nuestro dispositivo o quedarnos sin memoria libre. Además, dado que la aplicación puede seguir funcionando en segundo plano, este punto se vuelve aún más importante ya que un uso ineficiente de memoria podría tener consecuencias bastante graves.

Por ello, se han garantizado unos límites de memoria y cpu consumidas por nuestra aplicación que mostramos a continuación en la siguiente gráfica.

Los datos han sido obtenidos cuando nuestra aplicación estaba corriendo en un dispositivo Nexus 7 utilizando todas las funcionalidades que nos ofrece la aplicación y teniendo además el envio de datos al servidor funcionando en segundo plano.

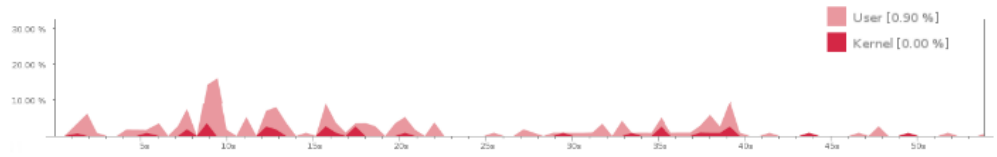

Figura 4.20: Uso de CPU

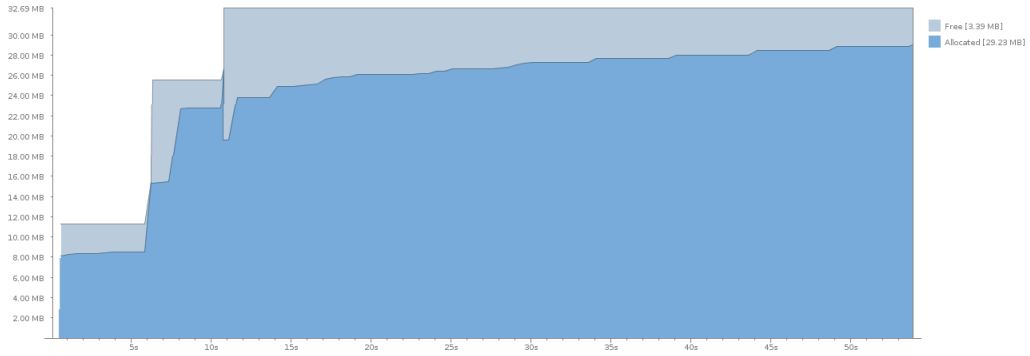

Figura 4.21: Uso de memoria

Con respecto al consumo de cpu, podemos ver como no llegamos ni al diez por ciento en el kernel y en términos de memoria no superamos los treinta megabytes a máximo rendimiento. Por ello podemos concluir que el uso de InfoWifiApp no afecta gravemente al rendimiento de nuestro dispositivo.

Chapter 5

Conclusions

The findings of this study and the future work lines will be presented in this chapter.

5.1. Project conclusions

As we wrote in introduction chapter, the goal of this project is develop an user interface which allow us to know all the information about wireless network which an Android device could detect.

To get this goal, we have a collect data system from wireless networks which shows the user collected information using plain text and graphics. Within this information a common user can choose what is the best network for him because it possible to know all the parameters of the network. Also, we provide data analytics to compare available networks.

On the other hand, we designed a communication system between our mobile and a server whose function is store the collected data sent by the device. On the device we have an implementation which convert android data in JSON object. We need to do this transformation because we are going to communicate two different devices with their respectives programming languages (Android in the device, Php and Sql in the server).

Server side has a php script listening for new request and storing sent information in a configured database. In our case, we are using a document oriented database because the information that we are sending has a special structure.

In order to manage the database easily, we store all the data using a stamp which give us information about what information is storing (sender device and time). So when we want to search about information which sent "Nexus 7 at 17:12"we are not going to need to see all the data table entries.

After testing our application, we got a really fantastic results: Less than 10 % of kernel usage and less than 30Mbits of memory consumption. For this reason, we can say that our application is optimized in terms of CPU usage, memory consumption and therefore in battery consumption.

In addition to get the indispensable requirement, we have developed some improvements like graphical visualization of data or the possibility to use shell commands.

It is neccesary to say that the developed application satisfies the regulatory framework, and this means that there is the posibility to sell the application to the common users or the networks operators without any legal problems.

Besides, after show the application to our work team, the application had the aprobation from all the project supervisors.

But I would like to give you a personal view about this project:

Developing this application I have learned to program in Android and Php, to establish communications between different devices, to configure wireless access points and to discover the world of documents oriented databases.

In my work team, Android knoweledges were not using in their projects so there weren't people who could help me when I had problems with my progress but now, I am gratefully with this because with that situation I learned to learn, I learned to solve problems by myself and I learned what I am capable of.

Regarding to the project, I need to say that there is a big burden: The application only can run if we are using Android as operative system.

The fact of use MongoDB as database I think that is one of the strongest points of this project because as I said in the correspondant chapter, we can create our own data structures.

As a point against, I could say that we should have tested our application in a real web server and not in a local server because in the future our application will be running there. Also, I need to say that I would have liked to develop some of the extra features that I am going to cite you as future work lines in the next section.

5.2. Future work lines

I think that there are a number of aspects which could have been taken into account when we developed the application.

I believe that a really interesting extra feature would be a “tester connection functionality”. With this item we could check if a network is working properly throwing a quick connection or a ping request.

A “Reall Traffic Monitor”would be another feature that I would implement. We could use to know how many users are in a determinate network. This feature was proposed to the team but was impossible to develop because if we want to know how many users are using the network we would needed to be connected in the same network and have additional privileges. For this reason, this proposal was not feasible since this feature would slow down the main operation of our application.

As we discussed in the corresponding chapter, this project has been implemented under T-Labs conditions. One of them was that our application can not use any root privileges. From my point of view, I think that if we use root privileges we could get more information about wireless network and develop more extra features so maybe would be interesting continue in this way if we want to improve our application.

Capítulo 6

Referencias

[1] Jesús Tomás Gironés, *El Gran Libro de Android.* 3 edición: Marcombo, S.A, 2013.

[2] Vicente Javier Eslava Muñoz, *El nuevo PHP, Conceptos Avanzados.* 1 edición: Bubok Publishing, S.L, 2013.

[3] Steve Klabnik, Yehuda Katz, Dan Gebhardt, Tyler Kellen and Ethan Resnick, *Json Examples,* http://jsonapi.org/examples/, Último acceso: 24 de noviembre de 2015.

[4] IETF, *RFC 5416,* https://tools.ietf.org/html/rfc5416, Último acceso: 21 de septiembre de 2015.

[5] Auth Luis Ayala, *Android Studio Curso Básico: Aprenda paso a paso.* 1 edición: Createspace, S.A, 2015.

[6] Wilson da Rocha Fransa, *MongoDB Data Modeling.* 1 edition: Packt Pubilshing, S.L, 2015.

[7] José Enrique Amaro Soriano, *El Gran Libro de Programación Avanzada Con Android.* 1 edición: Marcombo, S.A, 2012.

[8] James F. Kurose, University of Massachusetts, Amherst Keith W. Ross, Polytechnic University, Brooklyn, *Computer Networking: A Top-Down Approach.* 6 edition: Pearson, S.L, 2013.

[9] Neil Reidron Seide, *Manual De Redes Inalámbricas (802.11 WI-*

FI). 1 edición: MCGRAW-HILL, S.L, 2004.

[10] Alexánder Borbón A., Walter Mora F., *Edición de textos científicos con LATEX.* 2 edición: Inkscape, Tikz y Presentaciones Beamer, S.A, 2014.

[11] Carlos E.Zerpa, *Guía metodológica para elaborar proyectos de investigación.* 1 Edition: SH, S.A, 2009

[12] Fernando Valderrama, *Mediciones y presupuestos.* 1 edición: Reverte, S.L, 2010.

[13] Christine Bresnahan and Richard Blum, *Linux Essentials.* 2 Edition: SYBEX, S.L, 2015

[14] Hans J. Einsiedler, Nico Bayer, Kay Haensge, Roman Szczepanski, Martin Kurze,Thorsten Rettig, Francisco Gonzalez-Garcia, Andreas Roos, Stephan Bergtand and Jose Ignacio Moreno, *Efficient Transmission of Smartphone Application Traffic in Wireless Access Networks.* DOI: 10.1109/MobileCloud.2014.35, 2014.

[15] Hans J. Einsiedler, Nico Bayer, Kay Haensge, Roman Szczepanski, Andreas Roos and Stephan Bergtand, *Context Information Controlled Operation of Mobile Wireless Access Networks for Resource Efficiency.* Conference: 11. Fachgespräch Ortsbezogene Anwendungen und Dienste (Location Based Application and Services), LBAS 2014, At Darmstadt, Germany.

[16] MongoDB Developers, *MongoDB Java 2.12 API,* http://api.mongodb.com/java/2.12/, Último acceso: 4 de octubre de 2015.

[17] Radiant Silver Lbs, *Tutorial: How to design Android UI/GUIs,* https://www.youtube.com/watch?v=E6c3DGnvefY, Último acceso: 12 de octubre de 2015.

[18] Python Developers, *Peticiones Http and Python,* http://docs.python-requests.org/es/latest/user/quickstart.html, Último acceso: 23 de octubre de 2015.

[19] David Ruiz Urraca, *Desarrollo de una aplicación móvil cliente-*

servidor, http://biblioteca.unirioja.es/tfe/R000001713.pdf, Último acceso: 17 de noviembre de 2015.

[20] Python software developers, *Python API,* https://docs.python.org/2/c-api/, Último acceso: 9 de diciembre de 2015.

[21] Android Developers, *Android SDK,* https://developer.android.com, Último acceso: 10 de enero de 2016.

[22] Jefatura del Estado, *Ley Orgánica 15/1999,* https://www.boe.es/buscar/act.php?id=BOE-A-1999-23750, Último acceso: 20 de enero de 2016.

[23] Pilar Navas, *Ciclo de vida de un proyecto,* http://www.spw.cl/proyectos/apuntes2/cap6.htm/, Último acceso: 9 de marzo de 2016.

[24] Karen Alexis Handl, *Aplicación práctica del Diagrama de Gant,* http://face.unt.edu.ar/web/iadmin/wp-content/uploads/sites/2/2014/12/Aplicacion-practica-Diagrama-de-Gantt.pdf/, Último acceso: 5 de abril de 2015.

[25] MichaelPage Ingenieros, *Estudio de remuneración,* http://www.michaelpage.es/sites/michaelpage.es/files/ingenieros2016.pdf, Último acceso: 20 de Mayo de 2016.

ANEXOS

Anexos A

Planificación del proyecto y presupuesto

En este anexo se presenta la planificación que se ha llevado para realizar el presente trabajo, indicando tanto las tareas como su secuencización y temporización.

A.1. Marco regulador

La aplicación desarrollada tiene que cumplir (y cumple) las disposiciones establecidas en la LOPD (Ley Orgánica de Protección de Datos). La Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal, publicada en el BOE núm. 298, de 14/12/1999 [22] y entrada en vigor desde 14/01/2000 tiene por objeto garantizar y proteger, en lo que concierne al tratamiento de los datos personales, las libertades públicas y los derechos fundamentales de las personas físicas, y especialmente de su honor e intimidad personal y familiar.

Además, como potenciales desarrolladores de una aplicación que puede ser comercializada, es necesario tener en cuenta que se deben cumplir una serie de aspectos legales adicionales que se analizan a continuación:

- **Derechos propios y de terceros**

  En caso de utilizar recursos externos que requieran una licencia adicional será necesario disponer de esta. Además, deberemos establecer las medidas necesarias para evitar posible copias, plagios o imitaciones de la aplicación a desarrollar. En nuestro caso no necesitaremos ninguna licencia adicional por ello con establecer las medidas de seguridad pertienentes será suficiente.

- **Uso de menores**

  Es necesario enfocar el uso de nuestra aplicación a un determinado público, sobre todo en caso de aplicaciones no recomendadas para menores de 14 años ya que existe una legislación de consumidores y usuarios que podría iniciar repercusiones legales en caso de no cumplir con la normativa pertinente. En nuestro caso esta restricción no existe ya que esta aplicación está destinada para usuarios de cualquier edad.

- **Licencia y condiciones de uso**

  Es necesario redactar una serie de licencias de uso y condiciones que el usuario debe aceptar para poder hacer uso de la aplicación para que el futuro consumidor no pueda reclamarnos un uso indebido de la aplicación. En nuestro caso no se utilizan licencias externas ni se limitan las condiciones de uso de la aplicación.

- **Información y permisos**

  En todo momento el usuario tiene que ser consciente de los permisos que se requieren para poder instalar la aplicación. En nuestra aplicación los requisitos están especificados tanto en el presente documento como en la propia aplicación

- **Markets**

  Para poder subir nuestra aplicación al mercado de aplicaciones (Android Market en nuestro caso) es necesario que nuestra aplicación cumpla con los condiciones que este imponga. Por ahora esto no se ha tenido en cuenta ya que no será comercializada hasta versiones futuras.

- **Política de cookies**

  Dado que en nuestra aplicación no utilizamos cookies, no es necesario informar de estas al usuario.

A.2. Planificación

A.2.1. Ciclo de vida del proyecto

El conjunto de fases en las que se ha organizado el presente proyecto desde su inicio hasta su cierre han seguido un modelo en cascada en el que antes de poder avanzar a la siguiente etapa era necesario haber finalizado completamente la anterior. A continuación se presenta un esquema de las fases seguidas:

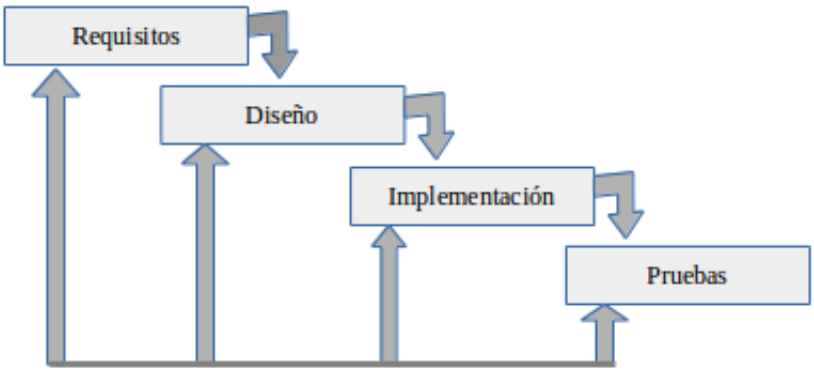

Figura A.1: Fases ciclo de vida

- **Requisitos**

  Mediante la obtención de requisitos se definen las necesidades y deseos del cliente en relación a la consecución del proyecto.

- **Diseño**

  Una vez tenemos los requisitos establecidos, se estructura el sistema a implementar identificando las partes de las que estará formado y la forma en la que se relacionan.

- **Implementación**

  Siguiendo el diseño establecido se implementa el código fuente.

- **Pruebas**

  Se comprueba que el producto cumple con los objetivos previstos.

A.2.2. Descomposición en tareas

Las tareas que se han seguido para la consecución del presente proyecto se pueden agrupar en cinco grandes bloques en los que se agrupan aquellas que pertenecen a la misma fase.

1. **Análisis**

   Subtareas:

   a) Planteamiento del problema

   b) Propuesta de la solución

c) Elección de entorno de desarrollo

Dedicación de personal:

- Jefe de proyecto: 64 horas –>0.37 hombres mes
- Analista programador: 48 horas –>0.28 hombres mes

2. **Diseño**

Subtareas:

a) Diseño del sistema completo e interfaz gráfica

Dedicación de personal:

- Jefe de proyecto: 55 horas –>0.32 hombres mes
- Analista programador: 72 horas –>0.41 hombres mes

3. **Implementación**

Subtareas:

a) Implementación del sistema de recogida de datos
b) Implementación del sistema de análisis de datos
c) Implementación de la comunicación con el servidor
d) Implementación de la interfaz gráfica
e) Implementación de funcionalidades extra

Dedicación de personal:

- Jefe de proyecto: 169 horas –>0.97 hombres mes
- Analista programador: 656 horas –>3.73 hombres mes
- Encargado de pruebas: 72 horas –>0.41 hombres mes

4. **Pruebas**

Subtareas:

a) Pruebas funcionales
b) Análisis de rendimiento

Dedicación de personal:

- Jefe de proyecto: 20 horas –>0.12 hombres mes
- Encargado de pruebas: 104 horas –>0.60 hombres mes

5. **Documentación**

   Subtareas:

   a) Elaboración de documentación

   Dedicación de personal:

   - Jefe de proyecto: 44 horas –>0.25 hombres mes
   - Analista programador: 104 horas –>0.60 hombres mes

A.2.3. Diagrama de Gant

Para realizar el seguimiento y control del progreso de cada una de las etapas del proyecto se ha seguido el siguiente diagrama de Gant:

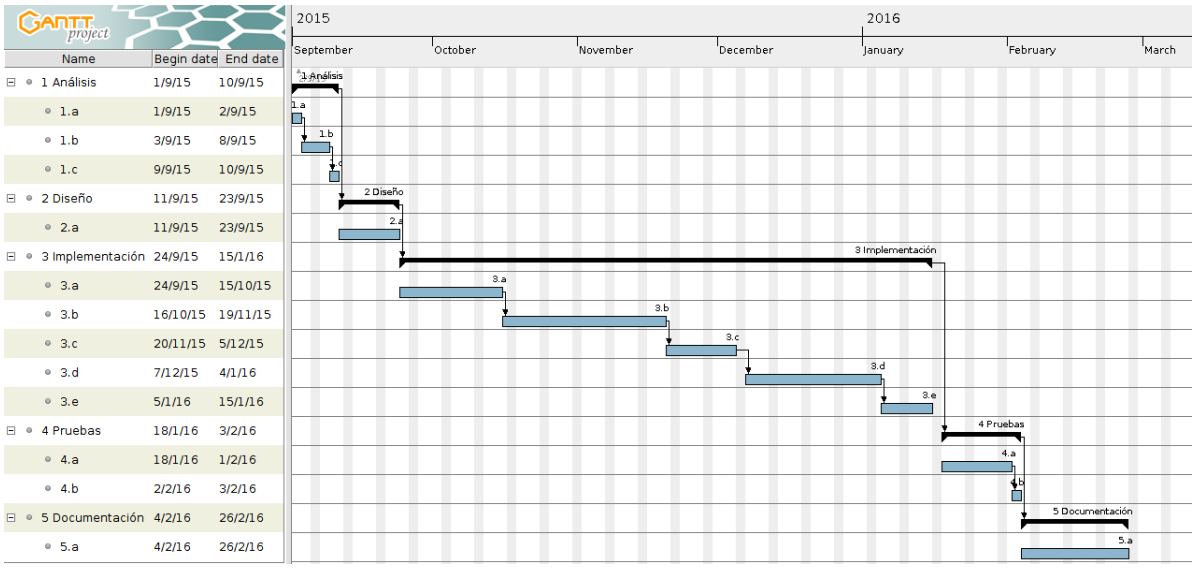

Figura A.2: Diagrama de Gant

A.3. Presupuesto

A continuación se presenta el presupuesto del presente proyecto desglosado en todos sus conceptos. Además, se incluye un cálculo de la amortización del producto para conocer a partir de qué punto empieza a ser rentable nuestro producto en caso de la comercialización de este.

A.3.1. Costes materiales

Los costes materiales representan los gastos generados por la compra de los materiales necesarios para la realización del proyecto. En nuestro caso se trata de un ordenador personal, una tablet y un dispositivo móvil.

Dado que todo el material utilizado pertenecía a los laboratorios de investigación, no considero aportuno incluir el costo de estos elementos en su totalidad ya que van a ser reutilizados para futuros proyectos. Por este motivo se calcula una amortización por el tiempo de uso que se le han dado indicada en la tabla A.1 como coste imputable. Este coste imputable se obtiene tras el uso de la siguiente fórmula:

$$\mathbf{Y = (M/D)\ x\ C\ x\ U}$$

Donde Y representa el coste imputable, M el número de meses utilizado, D la depreciación en meses, C el coste total en euros y U el uso en tanto por uno.

Es necesario indicar que todo el software que se ha utilizado ha sido libre luego no ha generado costes extra por licencias.

En todas las cantidades que se mencionan a continuación estará incluido el impuesto sobre el valor añadido (IVA).

Concepto	Cantidad	Coste/unidad (euros)	% Uso dedicado en el proyecto	Dedicación en proyecto (meses)	Periodo de depreciación (meses)	Coste imputable (euros)
Lenovo L2251p TFT 22"	1	83.49	100	6	60	8.35
Acer Aspire Revo r3700	1	391.00	100	6	60	39.10
Nexus 5 16GB	1	299.00	40	3	60	5.98
Nexus 9 32GB LTE+ WIFI	1	479.00	100	6	60	47.90
					Total:	101.33

Tabla A.1: Costes materiales

A.3.2. Costes humanos

Para la estimación de costes humanos [25] se ha establecido una jornada de trabajo de 8 horas diarias considerando que el mes tiene cuatro semanas con una totalidad de 22 días laborales. Por consiguiente, para nuestro proyecto 1 hombre mes equivale a 176 horas laborales (8 horas/día x 22 días).

Categoría	Dedicación hombres mes[1]	Coste hombre hora (euros)	Coste hombre mes (euros)	Coste (euros)
Jefe de Proyecto	2	35.00	6160.00	12320.00
Analista programador	5	24.00	4224.00	21120.00
Encargado de pruebas	1	24.00	4224.00	4224.00
			Total	37664.00

Tabla A.2: Costes humanos

A.3.3. Resumen de costes

El resumen de costes presenta el costo total del proyecto, por ende incluye tanto los costes humanos como los materiales.

Concepto	Total (euros)
Costes humanos	37664.00
Costes materiales	101.33
Total	37765.33

Tabla A.3: Resumen de costes

A.3.4. Amortización del producto

Por último se incluye un estudio en el que se calcula el número mínimo de ventas de la aplicación desarrollada que sería necesario para amortizar los gastos implicados en caso de decidir la comercialización de esta.

Coste Total (euros)	Inscripción en GooglePlay (euros)	Precio App (euros)	% ventas GooglePlay	Beneficio por venta (euros)	Ventas necesarias
37765.33	20	1.99	30	1.393	27126

Tabla A.4: Amortización

Anexos B

Summary

Smartphones have become in an indispensable devices in our daily life. Actually, more than 90 % of the world population has at least, one mobile device. The reason is easy to understand, we can use our mobile to do whatever we think, wherever and whenever: We can make a call, check our social profiles or pay our bill.

As we could think, to make this possible is necessary to have an internet connection in our device. We have two posibilities: Use a data mobile plan or use a wireless network which be available in our location.

Probably, many years ago we could have problems to find wireless networks because the internet access was not feasible for everyone. But now is really difficult to find public places like shopping centers, barbershops or a simply restaurant without wireless networks which get us an internet access.

So really, we are surrounded by wireless networks which bring us the possibility to have an internet connection.

But sometimes is difficult to choose which one is the best or what is the safest network.

Also, we need to know that today, mobile devices are very powerfull and have a lot of information available coming from different sources, e.g. sensors, APPs, networks, etc. However, this information is currently not used by network operators in order to optimize their networks.

So we have just to see two problems, the first one is the difficult choosing the best network for us and the other one is that we have a lot of information in our devices which are not being used by networks operators in order to improve their services.

For this reason, in cordination with T-Labs, I decided to create a graphical interface which allow us to answer these questions. This means that within this project we are going to develop an Android application which helps the

common users to choose the best network and helps the networks operators to improve their infrastructure.

The following scenario was proposed:

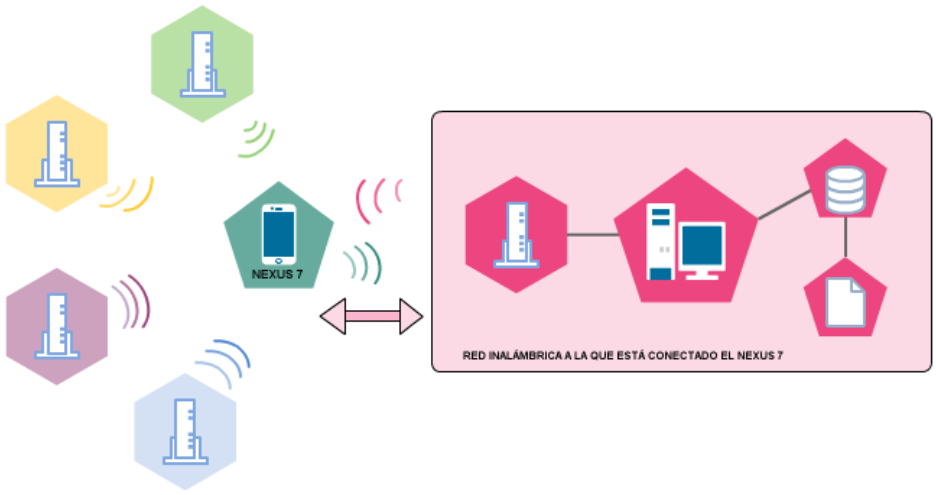

Figura B.1: Proposed scenario

As we can see in the previous image, our device (Nexus 7) is receiving signals from different access point and at the same time is connected to a local wireless network whose source is the computer which is being used as server and database as well.

At this point, we note that we are going to communicate different devices. Our application will be implemented in Android (Java) and the server will have a php script running listening for new requests. So we will need something that get us the possibility to have a successfull communication. To make this possible we are going to use JSON as intermediate language. This means that when we send the data to the server, we are going to create JSON Object as you can see below:

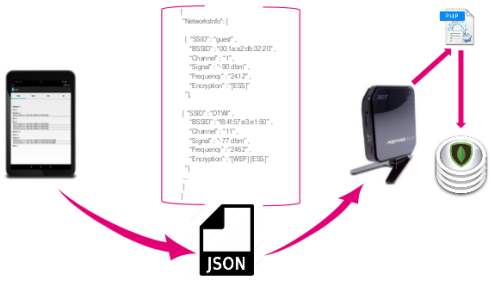

Figura B.2: Communication environment

In order to develop the application, we will need to have an implementation of hierarchical context collection system which allows to efficiently collect the relevant context information, an implementation of an interface management system which give us an analysis of the results and an implementation of a communicate system with local or web server.

Once we got familiar with Android development, we started researching about what kind of wireless information could be possible to get using an Android not rooted device. After this, we started developing a data collection system which offer us all the information about wireless networks which a device could detect. To help a common user to choose the best network, we analyse all these information showing it textually and graphically.

Now was the time to design the communication system between our device and the server (local server in our case). With this implementation we could send the collected information to a smart server which create statistics and improve the service that networks operators are offering to their clients.

When the elementary functions were implemented, was the turn of build the user interface and develop extra features like terminal emulator.

The project was completed after testing all the feautures and writing the necessary documentation.

At the end, we had an Android application which collect information about wireless networks, analyse that information and give us the possibility to have a communication between our device and a server. Down below you can see some InfoWifiApp screenshots:

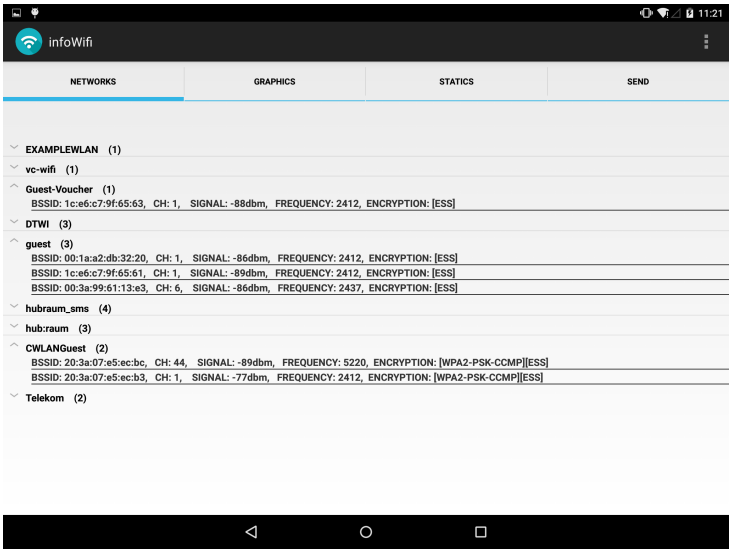

Figura B.3: InfoWifiApp scanning

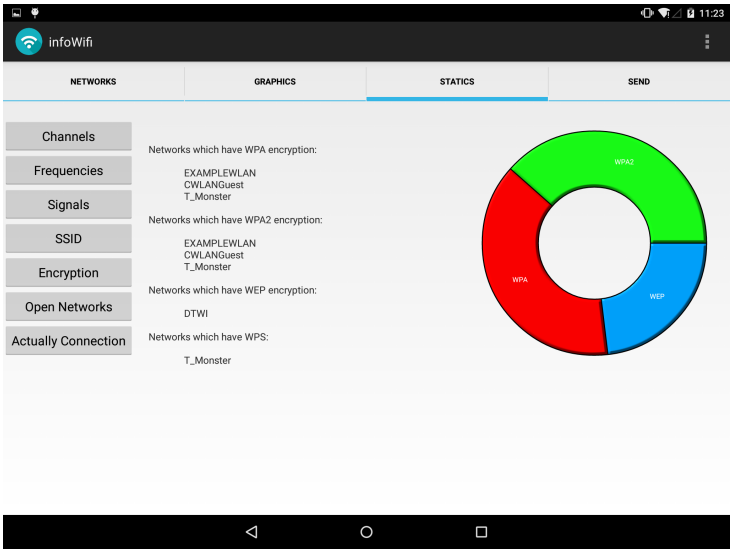

Figura B.4: InfoWifiApp getting statistics

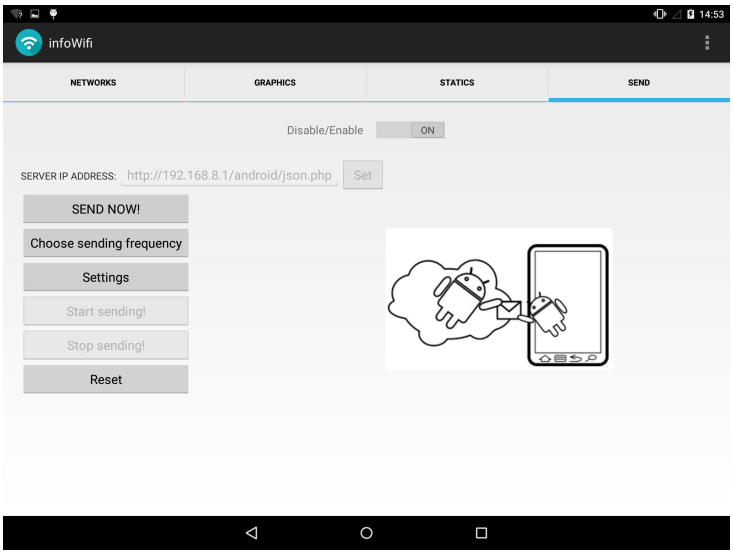

Figura B.5: InfoWifiApp sending collected data

Now we can see that within this application we have solved the two problem mentioned before, we help a common user choosing the appropriate network and if we think like a network operator and we implement an analytics system in the server, we could detect bugs, have information about what devices are using our network or information about how the final user is receiving our service.

What is striking is the posibility to use InfoWifiApp with any Android device which has a working Wifi module activated. For this reason, the user interface has been designed in a manner that the application adapts the views to the device which are using the application.

But this is not all, we designed our application in such a way that we can run InfoWifiApp in background, this means that we can be collecting data and sending the information periodically at the same time that we are listening music or we are writing a text message.

This feature need to be perfectly optimized because in the case that we do not do, we could see problems with our battery life. For this reason, for example, when we are collecting information about wireless networks, we only refresh the screen when something happens like when we found a new wireless network or when the signal strength changes. Anyway, we can choose refresh whenever we want using refresh option in settings. Also, if we are worry about our battery life but we need to send information to the server, we can choose sending frequency or do it manually.

If you are a Linux user, you should know that shell terminal is one of the best feautures of it: we can use terminal for all, we can do whatever we could imagine. The heart of Android is Linux so when we were designing this application we thought about make some shell terminal which bring us more information about wireless networks. After make it, we realized that really, this shell only give us additional information if we use a rooted device. As we can not use a rooted device (T-Labs conditions), this feature is not really usefull to our application but it will be really interesting in the future when we can use root privileges in the application.

In our case, we were using our laptop as a local server. To store the collected data, after research about what could be the best database to use, we decided to use MongoDB, a document oriented database. The choosing reason of this database is that we do not have to translate between objects in our application and SQL queries: we can just turn the object model directly into a document and unstructured data can be stored easily.

But we went further and we decided to store the information intelligently using a stamp to locate information easily in the future. This stamp was made with information about sender device: model device and sending time.

So when we want to locate or classify some information we only need know the model device or the time when the device sent the information.

In addition, probably, sometimes we will want to change the server location, but this is not a problem for us because when we are in sending settings, we can write the path location of our server.

After testing our application, we got the following results:

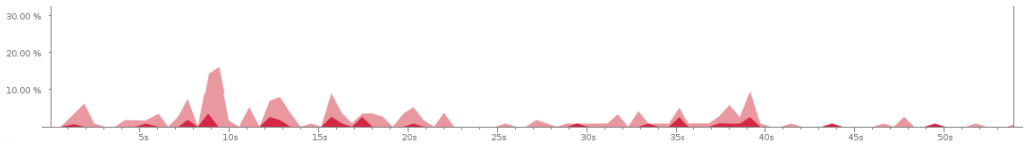

Figura B.6: CPU Performance results

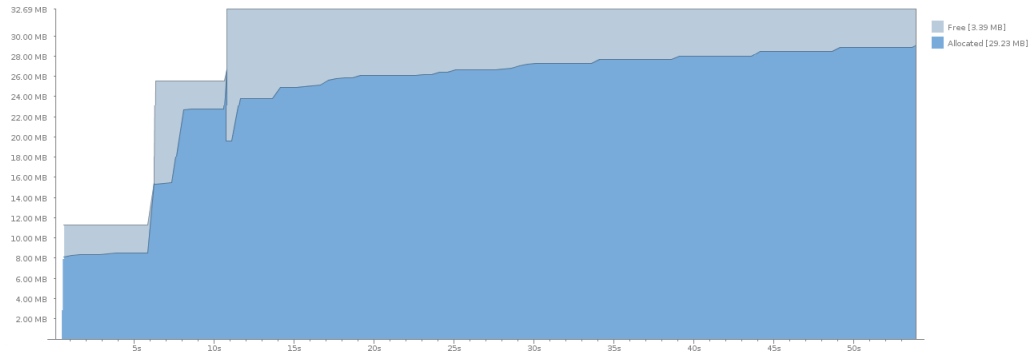

Figura B.7: Memory Performance results

As you can see in the graphics, we got a really fantastic results: Less than 10 % of kernel usage and less than 30Mbits of memory consumption. For this reason, we can say that our application is optimized in terms of CPU usage, memory consumption and therefore in battery consumption.

Also we should keep in mind that the developed application satisfies the regulatory framework so there is the posibility to sell the application to the common user (using online market or similar) or the networks operators without any legal problems.

After see the potential of Android, Php, Json and MongoDB I can say that we were right choosing these technologies and programming languages because we had all the necessary tools to develop InfoWifiApp.

Apart from this, I would like to propose some future work lines:

We could improve our application developing a “real traffic monitor”which show us how is the network in real time or implementing a “tester connection functionality”which bring us the posibility to know if the network is working properly.

If we use a rooted device, we could get more features so, maybe, this is an interesting future work line: use our application with root privileges.

Finally I need to say that this project was so important for me for different reasons: First one is that this was my first big project, second one is that I had to learn new programming languages that I never used before and was a challenge for me. Last one is, for me, the most important thing: I learned to solve my problems by myself, to trust me and to use my knowledges to learn

new things.

This project was made during my Erasmus+ programme in T-Labs, Berlin. Right now I only can say that I am very gratefully to do this intership because I had the posibility to improve my English, learn German, meet new people, meet new cultures and grow profesionally and personally.